Infoblox

API Documentation

NIOS 6.1

Copyright Statements

© 2011, Infoblox Inc.— All rights reserved.
The contents of this document may not be copied or duplicated in any form, in whole or in part, without the prior written permission of Infoblox, Inc.

The information in this document is subject to change without notice. Infoblox, Inc. shall not be liable for any damages resulting from technical errors or omissions which may be present in this document, or from use of this document.

This document is an unpublished work protected by the United States copyright laws and is proprietary to Infoblox, Inc. Disclosure, copying, reproduction, merger, translation, modification, enhancement, or use of this document by anyone other than authorized employees, authorized users, or licensees of Infoblox, Inc. without the prior written consent of Infoblox, Inc. is prohibited.

For Open Source Copyright information, refer to the *Infoblox NIOS Administrator Guide*.

Trademark Statements

Infoblox, the Infoblox logo, Grid, NIOS, bloxTools, NetMRI and PortIQ are trademarks or registered trademarks of Infoblox Inc.

All other trademarked names used herein are the properties of their respective owners and are used for identification purposes only.

Company Information

Infoblox is located at:
4750 Patrick Henry Drive
Santa Clara, CA 95054-1851, USA

Web: *www.infoblox.com*
www.support.infoblox.com

Phone: 408.625.4200
Toll Free: 888.463.6259
Outside North America: +1.408.716.4300
Fax: 408.625.4201

Product Information

Hardware Models: Infoblox-250-A, -550-A, -1050-A, -1550-A, -1552-A, -1852-A, and -2000-A.

Document Number: 400-0353-000, Rev. A

Document Updated: April 21, 2011

Warranty Information

Your purchase includes a 90-day software warranty and a one year limited warranty on the Infoblox appliance, plus an Infoblox Warranty Support Plan and Technical Support. For more information about Infoblox Warranty information, refer to the Infoblox Web site, or contact Infoblox Technical Support.

Preface

The API documentation explains how to install Infoblox DMAPI packages. It also describes in detail the objects and methods for writing and running scripts.

This preface includes the following topics:3

- *Document Overview* on page 4
- *Related Documentation* on page 4
- *Customer Care* on page 5
  - *User Accounts* on page 5
  - *Software Upgrades* on page 5
  - *Technical Support* on page 5

Document Overview

Infoblox DMAPI packages and objects are thoroughly documented. You can access the documentation through the Infoblox GUI or directly from a browser in either a Windows or UNIX/Linux environment. Note that when you use a UNIX/Linux environment, you cannot open the API documentation through the Infoblox GUI. You can, however, open the documentation from a browser.

Use one of the following to access the API documentation:

- From Grid Manager, select the Help panel, and then click **Documentation -> API Documentation**
- In your browser window, type **https://<IP address of device>/api/doc/**

Related Documentation

Other Infoblox appliance documentation:

- *Infoblox NIOS Administrator Guide*
- *Infoblox IBOS Administrator Guide*
- *Infoblox CSV Import Reference*
- *Infoblox-500, Infoblox-1000 and Infoblox-1200 Quick Start*
- *Infoblox User Guide for the Infoblox-1050, 1550, and 1552 Appliances*
- *Infoblox User Guide for the Infoblox-500, 550 Appliance*
- *Infoblox Installation Guide for the Infoblox-550, -1050, -1550, and -1552 Appliances*
- *Infoblox Installation Guide for the Infoblox-550-A, -1050-A, -1550-A, and -1552-A Appliances*
- *Infoblox Installation Guide for the Infoblox-1852-A Appliance*
- *Infoblox Installation Guide for the Infoblox-250 Appliance*
- *Infoblox Installation Guide for the Infoblox-250-A Appliance*
- *Infoblox Installation Guide for the Infoblox-2000 Appliance*
- *Infoblox Installation Guide for the Infoblox-2000-A Appliance*
- *Quick Start Guide for Installing vNIOS Software on Riverbed Services Platforms*
- *Quick Start Guide for Installing vNIOS Software on Cisco Application eXtension Platforms*
- *Infoblox Installation Guide for vNIOS Software on VMware*
- *Quick Start Guide for Installing vIBOS Software on VMware Platforms*
- *Infoblox Safety Guide*

To provide feedback on any of the Infoblox technical documents, please e-mail *techpubs@infoblox.com*.

Customer Care

This section addresses user accounts, software upgrades, licenses and warranties, and technical support.

User Accounts

The Infoblox appliance ships with a default user name and password. Change the default `admin` account password immediately after the system is installed to safeguard its use. Make sure that the NIOS appliance has at least one administrator account with superuser privileges at all times, and keep a record of your account information in a safe place. If you lose the `admin` account password, and did not already create another superuser account, the system will need to be reset to factory defaults, causing you to lose all existing data on the NIOS appliance. You can create new administrator accounts, with or without superuser privileges. For more information, refer to the *Infoblox Administrator Guide*.

Software Upgrades

Software upgrades are available according to the Terms of Sale for your system. Infoblox notifies you when an upgrade is available. Register immediately with Infoblox Technical Support at http://www.infoblox.com/en/support/product-registration.html to maximize your Technical Support.

Technical Support

Infoblox Technical Support provides assistance via the Web, e-mail, and telephone. The Infoblox Support web site at http://www.infoblox.com/en/support/support-center-login.html provides access to product documentation and release notes, but requires the user ID and password you receive when you register your product online at: http://www.infoblox.com/en/support/product-registration.html.

Infoblox - API Installation and Usage Instructions

- NAME
- VERSION
- Infoblox API Installation Instructions
  - Windows Management System
    - Requirements for Windows Management Systems
    - 1. Install Crypt::SSLeay and libwww-perl.
    - 2. PPM 3.x: Add the Infoblox device as a repository.
    - 3. PPM 3.x: Search the Infoblox device for Infoblox packages.
    - 4. PPM 3.x: Install Infoblox packages on your management system.
    - 5. PPM 4.0: Fetch and install Infoblox packages.
  - Unix Management System
    - Requirements for Unix Management Systems
    - DIRECT INSTALLATION
    - CPAN INSTALLATION
      - First Time CPAN Installation
      - 1. Install Crypt::SSLeay and CPAN::Site on your management system.
      - 2. Run CPAN::Site, instead of CPAN.
      - 3. Add the Infoblox device URL to the top of the CPAN URL list.
      - 4. (Optional) Reduce the number of CPAN fetch methods.
      - 5. Reload the CPAN::Site index.
      - 6. Install the Infoblox package on your management system.
- Infoblox API Usage Guidelines
  - UTF8 Considerations
  - `search()` Considerations
- Available Infoblox API Objects

NAME

Infoblox - API Installation and Usage Instructions

VERSION

```
6.1.0-121546
```

Infoblox API Installation Instructions

Infoblox provides a Perl API (application programming interface) to help facilitate the integration of the Infoblox device into network environments. The Infoblox DMAPI (Data and Management API) is a set of packages delivered with the Infoblox device to install the package. Use the usual Perl module installation tools on your management system to install the package. For Windows systems using the ActiveState build of Perl, the package is called PPM ("Perl Package Manager" or "Programmer's Package Manager"). For UNIX systems, the package is called CPAN, a global archive of Perl resources.

Windows Management System

Unix Management System

Windows Management System

To install the Infoblox DMAPI packages on a Windows management system, download and install PPM (from http://www.activestate.com/). Then do the following:

1. Install Crypt::SSLeay and libwww-perl.

2. PPM 3.x: Add the Infoblox device as a repository.

3. PPM 3.x: Search the Infoblox device for Infoblox packages.

4. PPM 3.x: Install Infoblox packages on your management system.

5. PPM 4.0: Fetch and install Infoblox packages.

Requirements for Windows Management Systems

- Perl 5.8.8 or later

- For Windows management systems: the ActiveState version of Perl 5.8.8 or later with PPM 3.x or later

- Crypt::SSLeay version 0.51 or later (and its dependencies: ssleay32.dll and libeay32.dll)

- LWP::UserAgent version 5.813 or later

- XML::Parser

1. Install Crypt::SSLeay and libwww-perl.

1.0 Check your Perl version.

The Crypt::SSLeay package is Perl version sensitive. Perl version 5.8.x requires different packages than 5.10.x. To check the Perl version installed, enter the command: perl -v.

If Perl 5.8.x is installed, the command will return the following:

```
C:\>perl -v
This is perl, v5.8.8 built for MSWin32-x86-multi-thread...
```

If Perl 5.10.x is installed, the command will return the following:

```
C:\>perl -v
This is perl, v5.10.0 built for MSWin32-x86-multi-thread...
```

1.1 Install Crypt-SSLeay.

The Crypt::SSLeay package allows you to make an HTTPS connection to the Infoblox device. Crypt::SSLeay is not part of the core Perl distribution. To check if it is installed, enter the command: ppm query Crypt-SSLeay. If the Crypt-SSLeay package is already installed, the query returns the following result:

```
C:\>ppm query Crypt-SSLeay
Querying target 1 (ActivePerl 5.8.8.817)
1. Crypt-SSLeay [0.51] OpenSSL glue that provides LWP https
```

If the Crypt::SSLeay package is not installed, the query returns the following:

```
C:\>ppm query Crypt-SSLeay
Querying target 1 (ActivePerl 5.8.8.817)
No matches for 'Crypt-SSLeay; see 'help query'.
```

To install Crypt::SSLeay (and its dependencies ssleay32.dll and libeay32.dll), use the following PPM command:

If the Perl version is 5.8.x:

```
C:\> ppm install http://theoryx5.uwinnipeg.ca/ppms/Crypt-SSLeay.ppd
```

If the Perl version is 5.10.x:

```
C:\> ppm install http://cpan.uwinnipeg.ca/PPMPackages/10xx/Crypt-SSLeay.ppd
```

1.2 Install libwww-perl.

If you need to update LWP::UserAgent, then you will have to install the latest libwww-perl package.

If the PPM version is 3.x, you will first need to add a PPM repository:

```
C:\> ppm repo add bribes http://www.bribes.org/perl/ppm
```

To install libwww-perl:

```
C:\> ppm install libwww-perl -force
```

2. PPM 3.x: Add the Infoblox device as a repository.

The following steps add the Infoblox device as a repository in your PPM 3.x configuration, included with Active Perl from version 5.8.0.802 to 5.8.8.817.

(For PPM 4.0, included with ActivePerl 5.8.8.818 and later, see "5. PPM 4.0: Fetch and install Infoblox packages.")

2.1 Add a PPM repository to the configuration by issuing the ppm repository command;

* For <repo_name>, type a meaningful name for the repository, such as 'Infoblox-device'. * For <ip_addr>, type the IP address that you use for managing the Infoblox device or a resolvable domain name (for a grid, use the address or domain name of the grid master).

```
C:\> ppm repo add <repo_name> https://<ip_addr>/api/dist/ppm
Repositories:
[1] <repo_name>
[2] ActiveState Package Repository
```

(Note that the packages are connected via HTTPS, not HTTP. You only need to add the repository once.)

2.2 To ensure that you have the correct URL, open a Web browser and visit the URL you specified for the repository. If the URL is correct, you can see a directory listing.

3. PPM 3.x: Search the Infoblox device for Infoblox packages.

Start PPM from a command shell:

```
C:\> ppm
PPM - Programmer's Package Manager version 3.4.
Copyright (c) 2001 ActiveState Software Inc.  All Rights Reserved.

Entering interactive shell. Using Term::ReadLine::Perl as readline library.

Type 'help' to get started.

ppm>

Use PPM to find the Infoblox package:

ppm> search Infoblox
Searching in Active Repositories
1. Infoblox [4.9999990014709] Infoblox Data and Management API 4.x
```

4. PPM 3.x: Install Infoblox packages on your management system.

Install the Infoblox package with PPM:

```
ppm> install Infoblox
====================
Install 'Infoblox' version 4.9999990014709 in ActivePerl 5.8.8.817.
====================
Downloaded 167481 bytes.
...
Successfully installed Infoblox version 4.9999990014709 in ActivePerl 5.8.8.817.
```

5. PPM 4.0: Fetch and install Infoblox packages.

There is a bug in PPM version 4.0 which is included in ActivePerl 5.8.8.818 and 5.8.8.819 which prevents PPM 4.0 from accessing a module archive with https. This bug, and ActiveState's workaround is described at:

http://bugs.activestate.com/show_bug.cgi?id=52508

If you have ActivePerl 5.8.8.818 or 5.8.8.819 (use perl -v at a command line to see), either upgrade to ActivePerl 5.8.8.820 or use the workaround described in the bug: http://bugs.activestate.com/show_bug.cgi?id=52508

You can also work around the bug by following the steps in section 5.3.

If you have ActivePerl 5.8.8.820 installed, skip the patch step in 5.1 and immediately install the package by following the steps in section 5.2.

5.1 PPM 4.0: Patch lib/ActivePerl/PPM/limited_inc.pm

If you have ActivePerl 5.8.8.818 or 5.8.8.819, please patch your PPM installation so PPM will be able to find modules stored in repositories accessed by https.

As shown in

http://bugs.activestate.com/show_bug.cgi?id=52508

apply this patch one time to your ActiveState PPM installation for each win32 client:

```
Index: lib/ActivePerl/PPM/limited_inc.pm
--- lib/ActivePerl/PPM/limited_inc.pm.~1~ Tue Nov 28 16:01:09 2006
+++ lib/ActivePerl/PPM/limited_inc.pm Tue Nov 28 16:01:09 2006
@@ -9,7 +9,7 @@
(my $dir = __FILE__) =~ s,/ActivePerl/PPM/.*,,;
Make a new directory, naming it something like "ibpack", and then change the command prompt to that
directory.
if (grep $_ eq $dir, @INC) {
- shift(@INC) while $INC[0] ne $dir;
+ push(@INC, shift(@INC)) while $INC[0] ne $dir;
}

1;
```

To find the full path to this file on your system, use perl -V at the command line. The contents of @INC are at the end of the output. The limited_inc.pm file is in the lib directory, not in the site/lib directory.

If you are unfamiliar with diffs, the + and - notation just means to replace the line:

```
shift(@INC) while $INC[0] ne $dir;
```

line with the

```
push(@INC, shift(@INC)) while $INC[0] ne $dir;
```

When complete, the lines of the patched limited_inc.pm should look like:

```
if (grep $_ eq $dir, @INC) {
# patch from http://bugs.activestate.com/show_bug.cgi?id=52508
#    shift(@INC) while $INC[0] ne $dir;
push(@INC, shift(@INC)) while $INC[0] ne $dir;
}
```

Continue on to step 5.2

5.2 PPM 4.0: Install the Infoblox modules

Enter the following command, substituting the IP address or resolvable hostname of the Infoblox device for <ip_addr>:

C:\tmp>ppm install https://<ip_addr>/api/dist/ppm/Infoblox.ppd

or you can use the GUI version of PPM by running the GUI, searching for the Infoblox module and installing it in the usual way.

5.3 PPM 4.0 Alternate Work Around to Install

Make a new directory and name the directory, for example, "ibpack". Then change the command prompt to that directory.

Make a new directory, naming it something like "ibpack", and then change the command prompt to that directory.

```
C:\> md ibpack

C:\> cd ibpack

C:\ibpack>
```

Next, transfer the files that contain the module distribution from the Infoblox device to your management system using 'lwp-request'. You must transfer two files: a description file, and the main distribution. The description file is always named "Infoblox.ppd".

Enter the following command, substituting the IP address or resolvable hostname of the Infoblox device for <ip_addr>:

```
C:\ibpack> lwp-request https://<ip_addr>/api/dist/ppm/Infoblox.ppd > Infoblox.ppd
```

View the Infoblox.ppd file:

```
C:\ibpack> type Infoblox.ppd
<?xml version="1.0" encoding="UTF-8"?>
<SOFTPKG NAME="Infoblox" VERSION="4,999999001478,0,0">
<TITLE>Infoblox</TITLE>
<ABSTRACT>Infoblox Data and Management API 4.x</ABSTRACT>
<AUTHOR>Infoblox, Inc.</AUTHOR>
<IMPLEMENTATION>
<CODEBASE HREF="Infoblox-4.999999001478.tar.gz" />
</IMPLEMENTATION>
</SOFTPKG>
```

The file name of the main distribution package is contained in the HREF attributes of the CODEBASE element. In the output shown above, it is "Infoblox-4.999999001478.tar.gz". Use lwp-request to fetch that file.

```
C:\ibpack> lwp-request https://<ip_addr>/api/dist/ppm/Infoblox-4.999999001478.tar.gz > Infoblox-
4.999999001478.tar.gz
```

Use PPM to install the Infoblox DMAPI from the local file system:

```
C:\ibpack> ppm install Infoblox.ppd
Unpacking Infoblox-4.999999001478...done
Generating HTML for Infoblox-4.999999001478...done
Installing to site area...done
64 files installed
```

The installation is complete.

Uninstalling and Reinstalling the Infoblox Package

After you configure PPM to search the Infoblox repository, use the appropriate command to uninstall and reinstall the Infoblox package:

```
ppm> uninstall Infoblox
====================
Remove 'Infoblox' version 4.9999990014709 in ActivePerl 5.8.8.817.
====================
...
Successfully removed Infoblox version 4.9999990014709 from ActivePerl 5.8.8.817.

ppm> install Infoblox
```

```
=====================
Install 'Infoblox' version 4.9999990014709 in ActivePerl 5.8.8.817.
=====================
...
Successfully installed Infoblox version 4.9999990014709 in ActivePerl 5.8.8.817.
```

Unix Management System

Requirements for Unix Management Systems

A working CPAN installation on the client UNIX machine when using CPAN::Site pre 1.02. Otherwise, a direct installation is required.

DIRECT INSTALLATION

To install the Infoblox DMAPI packages on a UNIX management system, first download and install the API package from:

```
https://<ip_addr>/api/dist/CPAN/authors/id/INFOBLOX/
```

where ip_addr is the IP address of the appliance. Then locate and download the file Infoblox-xxxxxxx.tar.gz where xxxxxxx is an integer depending on your API package version.

After you download the package, extract it to a temporary directory with:

```
tar xvfz Infoblox-xxxxxxx.tar.gz
```

Then execute the following commands:

```
cd Infoblox-xxxxxxx/
perl Makefile.PL
make
make install
```

Optionally, before you install, test the package by running:

```
make test
```

The installation is complete.

CPAN INSTALLATION

To install the Infoblox DMAPI packages on a UNIX management system, first download and install ActivePerl (from www.activestate.com), and then do the following steps:

1. Install Crypt::SSLeay and CPAN::Site on your management system.

2. Run CPAN::Site, instead of CPAN.

3. Add the Infoblox device URL to the top of the CPAN URL list.

4. (Optional) Reduce the number of CPAN fetch methods.

5. Reload the CPAN::Site index.

6. Install the Infoblox package on your management system.

Typically, the installation is done at the root level. See perlfaq8 under the question "How do I keep my own module/library directory?" for instructions on how to install a module as a user other than root.

If you have never used CPAN on the client UNIX machine, you will need to follow the instructions, when you run it for the first time, you will be prompted for many configuration variables. Follow the procedure at 0. First Time CPAN Installation

CPAN::Site Overview

The Infoblox modules are installed using a variant of CPAN named CPAN::Site. See

http://search.cpan.org/search?mode=module&query=CPAN%3A%3ASite

for more information on CPAN::Site.

In the usual CPAN model, there is a single CPAN repository, and every copy is a full and complete copy of every available package. In the CPAN::Site model, there are multiple repositories which may have a given module. As the Infoblox modules are deployed with CPAN::Site, you must use CPAN::Site instead of CPAN to install them. This is covered in more detail in step 2. If you use CPAN instead of CPAN::Site, typically the installation process will not find the Infoblox module in step 5.

One consequence of using CPAN::Site is the installation searches multiple sites for the same set of files, finding some of them in some locations but not in others. This is a normal result. Additionally, CPAN has the ability to search for and fetch files using several different mechanisms: curl, FTP, lynx, ncftpget, wget, and LWP. Both CPAN and CPAN::Site use each of these methods in turn when it tries to find files. If the CPAN configuration lists many sites in its URL list and many access methods, the installation may take a long time while CPAN::Site tries each of the access methods on each of the listed sites. For this reason, Infoblox recommends that you reduce the

number of sites in the URL list and the number of access methods as described in step 4.

If you are installing a new version of the Infoblox modules and your urllist configuration still includes the Infoblox appliance, skip directly to steps 5 and 6.

First Time CPAN Installation

If you have never used CPAN on this client machine before, enter the following command:

```
[root@lx2 ~]# perl -MCPAN -e shell
```

and follow the prompts through the CPAN configuration. You can accept all of the defaults, except for the list of CPAN sites close to your physical location. Navigate the menus and choose 1 or 2 CPAN sites that are close to your site. When your configuration choices are complete, exit CPAN by typing exit at the prompt.

1. Install Crypt::SSLeay and CPAN::Site on your management system.

The Crypt::SSLeay package allows you to make an HTTPS connection to the Infoblox device. If Crypt::SSLeay is not already installed, open a terminal window, login as the root user, and use CPAN to install the Crypt::SSLeay package and all its dependencies:

```
lx> perl -MCPAN -e 'install Crypt::SSLeay'
```

Install the CPAN::Site module:

```
lx> perl -MCPAN -e 'install CPAN::Site'
```

When the system prompts: "Running client or server?", enter 'client'. The installation process is asking if you want to install CPAN::Site as a server or as a client. Since CPAN::Site retrieves the Infoblox modules from a server, CPAN::Site acts as a client.

2. Run CPAN::Site, instead of CPAN.

Use CPAN::Site instead of CPAN:

```
lx> perl -MCPAN::Site -e shell
CPAN: File::HomeDir loaded ok

cpan shell -- CPAN exploration and modules installation (v1.87)
ReadLine support enabled
```

You must use CPAN::Site every time to access the Infoblox modules.

3. Add the Infoblox device URL to the top of the CPAN URL list.

3.0 You can manipulate the contents of the urllist with the usual perl functions push, pop, shift and unshift. push and pop operate on the end of the list, while shift and unshift operate on the beginning of the list. To empty the urllist, issue repeated shift or pop operands like this:

```
cpan> o conf urllist
urllist
[ftp://mirrors.kernel.org/pub/CPAN]
[http://mirrors.kernel.org/pub/CPAN]
Type 'o conf' to view all configuration items

cpan> o conf urllist shift

cpan> o conf urllist shift

cpan> o conf urllist
urllist
Type 'o conf' to view all configuration items
```

You can put an item on the list at the beginning with unshift and at the end of the list with push:

```
cpan> o conf urllist unshift ftp://mirrors.kernel.org/pub/CPAN

cpan> o conf urllist push ftp://bogus.site.at.end.of.list/CPAN

cpan> o conf urllist
urllist
[ftp://mirrors.kernel.org/pub/CPAN]
[ftp://bogus.site.at.end.of.list/CPAN]
Type 'o conf' to view all configuration items
```

When you finish configuring the urllist, you must have the Infoblox appliance as the first item in the list. This is a requirement of CPAN::Site. The best configuration of urllist is a two-item list with the Infoblox appliance first and one other close, fast CPAN site second. For example, at the Infoblox headquarters a good set of choices for urllist is:

```
cpan> o conf urllist
urllist
[https://<ip_addr>/api/dist/CPAN]
```

```
 [ftp://mirrors.kernel.org/pub/CPAN]
Type 'o conf' to view all configuration items
```

The complete list of CPAN sites can be found at:

http://www.cpan.org/SITES.html

3.1 Configure the Infoblox device first in the CPAN URL list, so that CPAN::Site finds it first when searching for packages. Type the IP address or domain name of your Infoblox device for <ip_addr> (for a grid, use the address of the grid master):

cpan> o conf urllist unshift https://<ip_addr>/api/dist/CPAN

(Note: You only need to add the repository to the URL list once.)

3.2 You must have at least one other normal CPAN site in the list in case your client needs other public packages. The following is a sample URL list for a set of CPAN sites in North America:

```
 cpan> o conf urllist
 urllist
 [https://<ip_addr>/api/dist/CPAN]
 [ftp://mirrors.kernel.org/pub/CPAN]
 [ftp://cpan.pair.com/pub/CPAN/]
Type 'o conf' to view all configuration items
```

If you do not have any other CPAN sites in your URL list, add at least one before continuing. Use the command

o conf urllist push <url_string>

3.3 To ensure that you have the correct URL for the Infoblox modules, open a Web browser and visit the URL you specified for the repository, in this example https://<ip_addr>/api/dist/CPAN. If the URL is correct, you can see a directory listing of the module files.

3.4 Save your CPAN configuration:

```
 cpan> o conf commit
commit: wrote '/usr/lib/perl5/5.8.8/CPAN/Config.pm'
```

4. (Optional) Reduce the number of CPAN fetch methods.

By reducing the number of CPAN fetch methods, you can shorten the installation process. For an explanation why Infoblox recommends this step, see the note in the introduction (before step 1).

CPAN (and CPAN::Site) supports up to six different ways to search for and fetch files: curl, FTP, lynx, ncftpget, wget, and LWP. Infoblox recommends that you choose the method that work best for you and only configure those methods. For the typical cases, Infoblox recommends keeping LWP, FTP, and ncftpget enabled and disabling curl, lynx and wget. To disable a fetch method, set it to an empty string and then commit the configuration. In the example below, you first display the configuration parameters for curl, lynx, and wget, and then set them to an empty string:

```
 cpan> o conf curl
 curl                              [/usr/bin/curl]

 cpan> o conf curl ""
 curl                              []

 cpan> o conf lynx
 lynx                            [/usr/bin/lynx]

 cpan> o conf lynx ""
 lynx                            []

 cpan> o conf wget
 wget                            [/usr/bin/wget]

 cpan> o conf wget ""
 wget                            []

 cpan> o conf commit
 commit: wrote '/usr/lib/perl5/5.8.8/CPAN/Config.pm'
```

5. Reload the CPAN::Site index.

Use CPAN to reload the CPAN::Site index:

```
 cpan> reload index
 CPAN: Storable loaded ok
 Going to read /root/.cpan/Metadata
 Database was generated on Sat, 09 Sep 2006 22:32:52 GMT
 CPAN: LWP::UserAgent loaded ok
 Fetching with LWP:
 https://<ip_addr>/api/dist/CPAN/authors/01mailrc.txt.gz
 ...
 Fetching with LWP:
 https://<ip_addr>/api/dist/CPAN/modules/03modlist.data.gz
```

```
LWP failed with code[404] message[Not Found]
Fetching with LWP:
ftp://mirrors.kernel.org/pub/CPAN/modules/03modlist.data.gz
Going to read /root/.cpan/sources/modules/03modlist.data.gz
Going to write /root/.cpan/Metadata
```

Note: Unlike CPAN, CPAN::Site is designed so that some of the files that describe the available packages are not found in all of the locations in your URL list. It is normal to see some "file not found" errors during the index reload.

6. Install the Infoblox package on your management system.

The usual CPAN process for installing Perl modules is:

6.1 Search

6.2 Test

6.3 Install

6.4 Uninstall and Reinstall

6.1 Search

Use the m operator to search for the Infoblox module:

```
cpan> m /Infoblox/
CPAN: Storable loaded ok
Going to read /root/.cpan/Metadata
Database was generated on Mon, 18 Sep 2006 06:29:55 GMT
CPAN: LWP::UserAgent loaded ok
...
Fetching with LWP:
https://<ip_addr>/api/dist/CPAN/site/01mailrc.txt.gz
Going to read /root/.cpan/sources/site/01mailrc.txt.gz
Module id = Infoblox
CPAN_USERID  INFOBLOX (Infoblox Technical Support <support@infoblox.com>)
CPAN_VERSION 4.9999990014709
CPAN_FILE    INFOBLOX/Infoblox-4.9999990014709.tar.gz
INST_FILE    (not installed)
```

If the search cannot find the Infoblox modules:

a. Ensure that you are using CPAN::Site, not CPAN as shown in 2. Run CPAN::Site, instead of CPAN. b. Check the Infoblox appliance in the urllist as shown in step 3.3.

6.2 Test

You can optionally test the package before the installation:

```
cpan> test Infoblox
Running test for module Infoblox
Running make for Infoblox-4.999990014709.tar.gz
...
All tests successful.
Files=1, Tests=1,  0 wallclock secs ( 0.74 cusr +  0.08 csys =  0.82 CPU)
/usr/bin/make test - OK
```

6.3 Install

Install the Infoblox package:

```
cpan> install Infoblox
Running install for module Infoblox
Running make for INFOBLOX/Infoblox-4.9999990014709.tar.gz
...
Writing /usr/lib/perl5/site_perl/5.8.8/i386-linux-thread-multi/auto/Infoblox/.packlist
Appending installation info to /usr/lib/perl5/5.8.8/i386-linux-thread-multi/perllocal.pod
/usr/bin/make install  -- OK
```

To exit the CPAN::Site shell:

```
cpan> quit
```

The installation is complete.

6.4 Uninstalling and Reinstalling the Infoblox Package

Although not directly supported by CPAN, it is simple to uninstall the Infoblox package using published techniques. One way to do it is provided by the modrm program that is described in the ExtUtils::Packlist documentation. See http://search.cpan.org/~yves/ExtUtils-Install-1.41/lib/ExtUtils/Packlist.pm#EXAMPLE for the modrm program. To reinstall it, repeat steps 5 and 6 above.

Infoblox API Usage Guidelines

The usage instructions are provided to assist you in using the Infoblox API.

UTF8 Considerations

If there are utf-8 characters in your script, remember to use the pragma 'use utf8'. If you need more information, please see http://search.cpan.org/~nwclark/perl-5.8.8/lib/utf8.pm on the utf8 pragma.

search() Considerations

When using the search() method, you could retrieve more information than expected. Include additional checks for exact matches.

Example

```
my @result_array = $session->search(
       "object" => "Infoblox::DNS::Record::A",
       "name" => "infoblox.com" );

my $obj;

foreach my $match (@result_array)
{
   if( ($match->name() eq "infoblox.com") && ($match->ipv4addr() eq "2.2.2.2") )
   {
      print "Matches\n";
      $obj = $match;
   }
}
```

Available Infoblox API Objects

Infoblox::Cursor - Manages paged searches.

Infoblox::DHCP::ExclusionRange - DHCP ExclusionRange object.

Infoblox::DHCP::ExclusionRangeTemplate - DHCP ExclusionRangeTemplate object.

Infoblox::DHCP::ExpandNetwork - Join or Expand an existing network

Infoblox::DHCP::FailOver - DHCP Failover object.

Infoblox::DHCP::Filter::MAC - DHCP MAC Address Filter object.

Infoblox::DHCP::Filter::MatchRule - DHCP Option Filter MatchRule object.

Infoblox::DHCP::Filter::NAC - DHCP NAC Filter object.

Infoblox::DHCP::Filter::Option - DHCP filter option object

Infoblox::DHCP::Filter::RelayAgent - DHCP relay agent filter object

Infoblox::DHCP::FilterRule::MAC - DHCP MAC Address Filter Rule object.

Infoblox::DHCP::FilterRule::NAC - DHCP NAC Filter Rule object.

Infoblox::DHCP::FilterRule::Option - DHCP filter rule option object.

Infoblox::DHCP::FilterRule::RelayAgent - DHCP RelayAgent Filter Rule object.

Infoblox::DHCP::FixedAddr - DHCP Fixed Address object.

Infoblox::DHCP::FixedAddrTemplate - DHCP Fixed Address template object.

Infoblox::DHCP::IPv6FixedAddr - DHCP IPv6 Fixed Address object.

Infoblox::DHCP::IPv6FixedAddrTemplate - DHCP IPv6 Fixed Address Template object

Infoblox::DHCP::IPv6Network - DHCP IPv6 Network object

Infoblox::DHCP::IPv6NetworkContainer - DHCP IPv6 Network Container object

Infoblox::DHCP::IPv6NetworkTemplate - DHCP IPv6 Network Template object

Infoblox::DHCP::IPv6OptionDefinition - DHCP IPv6 option definition object

Infoblox::DHCP::IPv6OptionSpace - DHCP IPv6 OptionSpace object.

Infoblox::DHCP::IPv6Range - DHCP IPv6 Range object.

Infoblox::DHCP::IPv6RangeTemplate - DHCP IPv6 RangeTemplate object.

Infoblox::DHCP::IPv6SharedNetwork - DHCP IPv6 Shared Network object

Infoblox::DHCP::Lease - DHCP Lease object.

Infoblox::DHCP::MAC - DHCP MAC object.

Infoblox::DHCP::MSOption - Microsoft (r) Server DHCP Option object.

Infoblox::DHCP::MSServer - Microsoft (r) DHCP Server object.

Infoblox::DHCP::MSSuperscope - Microsoft (r) Server Superscope object.

Infoblox::DHCP::Member - DHCP Member object.

Infoblox::DHCP::Network - DHCP Network object

Infoblox::DHCP::NetworkContainer - DHCP Network Container object

Infoblox::DHCP::NetworkTemplate - DHCP Network Template object

Infoblox::DHCP::Option - DHCP Option object.

Infoblox::DHCP::Option60MatchRule - DHCP Option60MatchRule object.

Infoblox::DHCP::OptionDefinition - DHCP option definition object

Infoblox::DHCP::OptionSpace - DHCP OptionSpace object.

Infoblox::DHCP::OrderedRanges - Ordered Ranges object.

Infoblox::DHCP::Range - DHCP Range object.

Infoblox::DHCP::RangeTemplate - DHCP RangeTemplate object.

Infoblox::DHCP::RoamingHost - DHCP Roaming Host object.

Infoblox::DHCP::SharedNetwork - DHCP Shared Network object

Infoblox::DHCP::SplitNetwork - Split network object.

Infoblox::DHCP::Statistics - Manages DHCP statistics.

Infoblox::DHCP::Template - DHCP Template object.

Infoblox::DHCP::View - DHCP Network View object

Infoblox::DNS::AllRecords - A synthetic object used to return all record object types belonging to a zone.

Infoblox::DNS::BulkHost - Create a group of hosts.

Infoblox::DNS::DnssecTrustedKey - DNSSEC trusted key object.

Infoblox::DNS::GlueRecordAddr - Infoblox DNS Glue Record Object.

Infoblox::DNS::Host - DNS Host record object.

Infoblox::DNS::MSServer - Microsoft (r) DNS Server object

Infoblox::DNS::Member - DNS Member object

Infoblox::DNS::NSEC3PARAM - NSEC3PARAM Record object

Infoblox::DNS::Nameserver - DNS Nameserver object

Infoblox::DNS::Nameserver::Address - NS Name Server object.

Infoblox::DNS::Record::A - DNS A record object.

Infoblox::DNS::Record::AAAA - DNS AAAA record object.

Infoblox::DNS::Record::CNAME - DNS CNAME record object.

Infoblox::DNS::Record::DNAME - DNAME record object.

Infoblox::DNS::Record::DNSKEY - DNSKEY Record object.

Infoblox::DNS::Record::DS - DS Record object.

Infoblox::DNS::Record::MX - DNS MX record object

Infoblox::DNS::Record::NAPTR - NAPTR record object.

Infoblox::DNS::Record::NS - DNS NS record object

Infoblox::DNS::Record::NSEC - NSEC Record object

Infoblox::DNS::Record::NSEC3 - NSEC3 Record object.

Infoblox::DNS::Record::PTR - DNS PTR record object.

Infoblox::DNS::Record::RRSIG - RRSIG Record object

Infoblox::DNS::Record::SRV - DNS SRV record object

Infoblox::DNS::Record::TXT - DNS TXT record object.

Infoblox::DNS::RootNameServer - Custom root name server object.

Infoblox::DNS::Ruleset - Ruleset object

Infoblox::DNS::Ruleset::NxdomainRule - Rule of Ruleset object

Infoblox::DNS::SRG - Create a shared record group.

Infoblox::DNS::SharedRecord::A - DNS shared A record object.

Infoblox::DNS::SharedRecord::AAAA - DNS AAAA shared record object.

Infoblox::DNS::SharedRecord::MX - DNS shared MX record object

Infoblox::DNS::SharedRecord::SRV - DNS shared SRV record object

Infoblox::DNS::SharedRecord::TXT - DNS shared TXT record object.

Infoblox::DNS::Sortlist - DNS Sortlist Object.

Infoblox::DNS::TSIGKey - DNS TSIGKey object

Infoblox::DNS::View - DNS View object

Infoblox::DNS::Zone - DNS Zone object

Infoblox::Grid - Manages the Grid settings.

Infoblox::Grid::Admin::AdAuthServer - Active Directory Authentication Server object

Infoblox::Grid::Admin::AdAuthService - Active Directory Authentication Service object

Infoblox::Grid::Admin::AuthPolicy - Authentication Policy object

Infoblox::Grid::Admin::Group - Admin Group object.

Infoblox::Grid::Admin::LocalUserAuthService - Local User Authentication Service object

Infoblox::Grid::Admin::Permission - Admin Permission object.

Infoblox::Grid::Admin::RadiusAuthServer - RADIUS Authentication Server object

Infoblox::Grid::Admin::RadiusAuthService - RADIUS Authentication Service object

Infoblox::Grid::Admin::Role - Admin Role object.

Infoblox::Grid::Admin::User - User Admin object.

Infoblox::Grid::CSVImportStatus - CSV import status object

Infoblox::Grid::DHCP - Grid DHCP services object.

Infoblox::Grid::DNS - DNS objects management and configuration

Infoblox::Grid::DNS::BulkHostNameTemplate - Manages the DNS bulk host name formats defined at the grid level.

Infoblox::Grid::DNS::DNS64Group - DNS64 Synthesis Group object

Infoblox::Grid::DNS::Nsgroup - Creates a name server group.

Infoblox::Grid::DateTime - Grid date and time object.

Infoblox::Grid::ExtensibleAttributeDef - Extensible Attribute Definition object.

Infoblox::Grid::ExtensibleAttributeDef::ListValue - Extensible Attribute Definition List Value object.

Infoblox::Grid::FileDistributionDir - Grid FileDistributionDir object.

Infoblox::Grid::GlobalSmartFolder - Global Smart Folder object.

Infoblox::Grid::IFMap - interface to configure the IF-MAP server object properties.

Infoblox::Grid::IFMap::Namespace - IF-MAP Namespace object

Infoblox::Grid::KerberosKey - Grid kerberos key object.

Infoblox::Grid::MSServer - Microsoft (r) Grid Server object.

Infoblox::Grid::MSServer::DHCP - Grid Microsoft (r) Server DHCP object.

Infoblox::Grid::MSServer::DNS - Grid Microsoft (r) Server DNS object.

Infoblox::Grid::Member - Grid Member object.

Infoblox::Grid::Member::BGP::AS - BGP (Border Gateway Protocol) Autonomous System (AS) object.

Infoblox::Grid::Member::BGP::Neighbor - BGP (Border Gateway Protocol) neighbor object

Infoblox::Grid::Member::CaptivePortal - Captive Portal object.

Infoblox::Grid::Member::CaptivePortal::File - Captive Portal File object

Infoblox::Grid::Member::DHCP - DHCP member object.

Infoblox::Grid::Member::DNS - Grid Member DNS object.

Infoblox::Grid::Member::FileDistribution - Grid Member FileDistribution object.

Infoblox::Grid::Member::Interface - Interface member object.

Infoblox::Grid::Member::License - License object.

Infoblox::Grid::Member::OSPF - OSPF (Open Shortest Path First) member object.

Infoblox::Grid::Member::ServiceStatus - Service Status object.

Infoblox::Grid::Member::StaticRoute - Static route member object

Infoblox::Grid::NTPAccess - Grid Network Time Protocol (NTP) access object.

Infoblox::Grid::NTPKey - Grid Network Time Protocol (NTP) key object.

Infoblox::Grid::NTPServer - Grid Network Time Protocol (NTP) server object.

Infoblox::Grid::NatGroup - NAT Group object.

Infoblox::Grid::PersonalSmartFolder - Personal Smart Folder object.

Infoblox::Grid::RecordNamePolicy - Manages the DNS record name policies defined at the grid level.

Infoblox::Grid::RootNameServer - Custom root name server object.

Infoblox::Grid::SNMP::Admin - Manages a SNMP Admin object.

Infoblox::Grid::SNMP::QueriesUser - Grid SNMP QueriesUser object.

Infoblox::Grid::SNMP::TrapReceiver - Grid SNMP TrapReceiver object.

Infoblox::Grid::SNMP::User - Grid SNMP User object

Infoblox::Grid::ScheduledBackup - Scheduled Backup object.

Infoblox::Grid::ScheduledTask - Scheduled Task object.

Infoblox::Grid::ScheduledTask::ChangedObject - Scheduled Task Changed Object.

Infoblox::Grid::SmartFolder::GroupBy - Smart Folder Group By object

Infoblox::Grid::SmartFolder::GroupByValue - Smart Folder Group By Value object

Infoblox::Grid::SmartFolder::QueryItem - Smart Folder Query Item object

Infoblox::Grid::SmartFolderChildren - Smart Folder retrieval item.

Infoblox::Grid::SyslogServer - Syslog server object.

Infoblox::Grid::TimeZone - Grid time zone object.

Infoblox::IFMap::AuthRule - IF-MAP Authorization Rules object.

Infoblox::IFMap::CACertificate - IF-MAP CA Certificate object.

Infoblox::IFMap::Client - IF-MAP Client object

Infoblox::IFMap::Client::Certificate - IF-Map Client Certificate object

Infoblox::IFMap::Client::Credentials - IF-MAP Credentials object

Infoblox::IFMap::Federation::Connector - IF-MAP Connector object

Infoblox::IFMap::Federation::Subscription - IF-MAP Subscription object

Infoblox::IFMap::Identifier - IF-MAP Identifier object

Infoblox::IFMap::Identifier::Attribute - IF-MAP Identifier Attribute object

Infoblox::IFMap::Identifier::Definition - IF-MAP Custom Identifier Type (CIT) Definition object

Infoblox::IFMap::Identifier::Definition::Attribute - IF-MAP custom identifier definition attribute object.

Infoblox::IPAM::Address - IPAM address object

Infoblox::IPAM::DiscoveryTask - Network discovery configuration

Infoblox::IPAM::DiscoveryTask::VServer - VMWare discovery server object.

Infoblox::IPAM::Statistics - Manages IPAM statistics.

Infoblox::IPAM::TCPPort - Network discovery TCP port

Infoblox::Session - Session object to manipulate data within a grid.

Infoblox::Cursor - Manages paged searches.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- METHODS
  - fetch_size( )
  - other parameters
  - Supported object types
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Cursor - Manages paged searches.

DESCRIPTION

This class enables users to conduct database searches returning large datasets in an object-by-object manner.

CONSTRUCTOR

```
my $cursor=$session->new_cursor(
                                fetch_size     => 200,
                                object         => "Infoblox::DNS::AllRecords",
                                return_methods => ['name'],
                                name           => ".*"
                               );
```

METHODS

This section describes all the methods that you can use when creating a cursor used for paged searches.

fetch_size( )

This method is used to set the amount of records returned from the back-end to the client for every request. Larger values will cause less requests to be sent to the back-end and more memory used on the client-side; smaller values will cause more requests.

The default value is 100.

other parameters

Other parameters for this constructor are any valid search parameters as described in Infoblox::Session->search()

Supported object types

Paged search is supported for the following object types:

Infoblox::DNS::AllRecords

SAMPLE CODE

The following sample code demonstrates how to execute a paged search.

```
#PROGRAM STARTS: Include all the modules that will be used
   $cursor=$session->new_cursor(
                                fetch_size => 100,
                                object => "Infoblox::DNS::AllRecords",
                                name   => '.*',
                               );

   while ($current_record = $cursor->fetch()) {
       # $current_record will contain a matching record object
   }
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->search()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::ExclusionRange - DHCP ExclusionRange object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::Range->exclude( )
- METHODS
  - comment( )
  - end_address( )
  - start_address( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::ExclusionRange - DHCP ExclusionRange object.

DESCRIPTION

The DHCP exclusion range describes the range of IP addresses that can be excluded in an address range. If static addresses are assigned to certain hosts in the middle of an address range, those addresses can be excluded from the address range so the DHCP server does not assign those IP addresses to clients.

CONSTRUCTOR

```
my $exclusion_range = Infoblox::DHCP::ExclusionRange->new(
      end_address   => $ipv4addr,      #Required
      start_address => $ipv4addr,      #Required
      comment       => $string,        #Optional / Default is empty
);
```

MODULE METHODS

The following functions are available to apply to a DHCP exclusion range object.

Infoblox::DHCP::Range->exclude( )

Use this function to specify exclusion range for the DHCP range. See Infoblox::DHCP::Range->exclude() for parameters and return values.

Example

```
#Construct a DHCP ExclusionRange object
my $exclusion_range = Infoblox::DHCP::ExclusionRange->new(
        end_address   => "10.0.0.8",
        start_address => "10.0.0.5",
        comment       => "Exclude four IP addresses"
);

# Configure DHCP ExclusionRange on the DHCP Range object
my $response = $dhcp_range->exclude([$exclusion_range]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP exclusion range object.

comment( )

Use this method to set or retrieve a descriptive comment of a DHCP exclusion range object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $exclusion_range->comment();
#Modify comment
$exclusion_range->comment("Modifying the DHCP exclusion range comment");
```

end_address( )

Use this method to set or retrieve the end address of a DHCP exclusion range object.

An end address is the last IP address in the DHCP exclusion range which is not available for the clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the end address of the DHCP exclusion range in IPv4 address format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get end_address
my $end_address = $exclusion_range->end_address();
#Modify end_address
$exclusion_range->end_address("10.0.0.9");
```

start_address( )

Use this method to set or retrieve the start address of a DHCP exclusion range object.

A start address is the first IP address in the DHCP exclusion range which is not available for the clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the start address of the DHCP exclusion range in IPv4 address format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get start_address
my $start_address = $exclusion_range->start_address();
#Modify start_address
$exclusion_range->start_address("10.0.0.6");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP exclusion range object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP exclusion range object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
```

```
                master   => "192.168.1.2", #appliance host ip
                username => "admin",       #appliance user login
                password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the Network object
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    comment => "add network",
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
    or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network object added to Infoblox appliance successfully\n";

#Create the DHCP Range object
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr   => "10.0.0.10",
        network    => "10.0.0.0/8",
        start_addr => "10.0.0.1",
);
unless($dhcp_range) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

#Add the DHCP Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to Infoblox appliance successfully\n";
```

#Create a DHCP exclusion range object

```
my $exclusion_range = Infoblox::DHCP::ExclusionRange->new(
         end_address   => "10.0.0.8",
         start_address => "10.0.0.5",
         comment       => "Exclude four IP addresses"
);
unless($exclusion_range) {
       die("Construct DHCP exclusion range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP exclusion range object created successfully\n";
```

#Add DHCP exclusion range to DHCP Range

```
#Get the DHCP Range object
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::Range",
    start_addr => "10.0.0.1"
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP Range object found at least 1 matching entry\n";

#Apply the changes to the DHCP Range object
$object->exclude([$exclusion_range]);

#Submit the changes to the Session
$session->modify($object)
```

```
    or die("Adding DHCP exclusion range to DHCP Range object failed: ",
           $session->status_code() . ":" . $session->status_detail());
 print "DHCP Range with exclusion range updated to Infoblox appliance successfully\n";
```

#Modify DHCP exclusion range

```
 #Modifying the end address of the DHCP exclusion range object.
 $exclusion_range->end_address("10.0.0.9");

 #Modifying the start address of the DHCP exclusion range object.
 $exclusion_range->start_address("10.0.0.6");

 #Modifying the comment of the DHCP exclusion range object.
 $exclusion_range->comment("Modified DHCP Exclusion Range");

 #Apply changes to the DHCP Range object.
 $object->exclude([$exclusion_range]);

 #Update DHCP Range object through the Infoblox session.
 $session->modify($object)
   or die("Modify DHCP Range object failed: ",
   $session->status_code() . ":" . $session->status_detail());
 print "DHCP Range object with modified DHCP exclusion range updated to Infoblox appliance
successfully\n";
```

#Remove DHCP exclusion range

```
 #Remove the exclusion range from the DHCP Range object.
 $object->exclude([]);

 #Update DHCP Range object through the Infoblox session.
 $session->modify($object)
   or die("Modify DHCP Range object failed: ",
   $session->status_code() . ":" . $session->status_detail());
 print "Removed DHCP exclusion range from DHCP Range successfully\n";
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(),Infoblox::Session->get(),Infoblox::Session->modify(),Infoblox::DHCP::Range,Infoblox::DHCP::Network

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::ExclusionRangeTemplate - DHCP ExclusionRangeTemplate object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::RangeTemplate->exclude( )
- METHODS
  - comment( )
  - number_of_addresses( )
  - offset( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::ExclusionRangeTemplate - DHCP ExclusionRangeTemplate object.

DESCRIPTION

The DHCP exclusion range template describes the range of IP addresses that can be excluded in an address range template. If static IP addresses are assigned to certain hosts in the middle of an address range, those addresses can be excluded from the address range so the DHCP server does not assign those IP addresses to the clients.

CONSTRUCTOR

```
my $exclusion_range_template = Infoblox::DHCP::ExclusionRangeTemplate->new(
      offset               => $num,          #Required
      number_of_addresses  => $num,          #Required
      comment              => $string,       #Optional / Default is empty
);
```

MODULE METHODS

The following functions are available to be applied to a DHCP exclusion range template object.

Infoblox::DHCP::RangeTemplate->exclude( )

Use this function to specify exclusion range for the DHCP range template object. See Infoblox::DHCP::RangeTemplate->exclude() for parameters and return values.

Example

```
#Construct a DHCP ExclusionRangeTemplate object
my $exclusion_range_template = Infoblox::DHCP::ExclusionRangeTemplate->new(
        offset               => "10",
        number_of_addresses  => "5",
        comment              => "Exclude five IP addresses"
);

# Configure DHCP ExclusionRangeTemplate on the DHCP Range template object
my $response = $dhcp_range_template->exclude([$exclusion_range_template]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP exclusion range template object.

comment( )

Use this method to set or retrieve a descriptive comment of a DHCP exclusion range template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $exclusion_range_template->comment();
#Modify comment
$exclusion_range_template->comment("Modifying the DHCP exclusion range template comment");
```

number_of_addresses( )

Use this method to set or retrieve the number of addresses in the DHCP exclusion range template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of addresses in this exclusion range template.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get number_of_addresses
my $number_of_addresses = $exclusion_range_template->number_of_addresses();
#Modify number_of_addresses
$exclusion_range_template->number_of_addresses("10");
```

offset( )

Use this method to set or retrieve the address offset of the DHCP exclusion range template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The address offset of this exclusion range template. The offset is calculated from the start address of network template to which the DHCP range template is assigned to. For example, if offset is set to 10 and number_of_addresses is set to 10, and network is set to 10.0.0.0/8, then start address of exclusion range created from this exclusion range template will be 10.0.0.10 and end address will be 10.0.0.20.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get offset
my $offset = $exclusion_range_template->offset();
#Modify offset
$exclusion_range_template->offset("10");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP exclusion range object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#PROGRAM STARTS: Include all the modules that will be used

```
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
```

```
                 password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the DHCP Range template object

my $dhcp_range_template = Infoblox::DHCP::RangeTemplate->new(
        name                    => "custom_range_template",
        offset                  => "10",
        number_of_addresses      => "10",
        authority               => "TRUE",
        bootfile                => "bootfile1.com",
);
unless($dhcp_range_template) {
       die("Construct DHCP Range template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range template object created successfully\n";

#Add the DHCP Range object into the Infoblox appliance through a session
$session->add($dhcp_range_template)
    or die("Add DHCP Range template object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to Infoblox appliance successfully\n";

#Create a DHCP exclusion range templat object

my $exclusion_range_template = Infoblox::DHCP::ExclusionRangeTemplate->new(
         offset              => "10",
         number_of_addresses => "5",
         comment             => "Exclude five IP addresses"
);
unless($exclusion_range_template) {
       die("Construct DHCP exclusion range template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP exclusion range template object created successfully\n";

#Add DHCP exclusion range template to DHCP Range template

#Get the DHCP Range template object
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::RangeTemplate",
    name       => "custom_range_template"
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP Range template object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP Range template object found at least 1 matching entry\n";

#Apply the changes to the DHCP Range template object
$object->exclude([$exclusion_range_template]);

#Submit the changes to the Session
$session->modify($object)
    or die("Adding DHCP exclusion range to DHCP Range template object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range template with exclusion range template updated to Infoblox appliance successfully\n";

#Remove DHCP exclusion range template

#Remove the exclusion range template from the DHCP Range template object.
$object->exclude([]);

#Update DHCP Range template object through the Infoblox session.
$session->modify($object)
  or die("Modify DHCP Range template object failed: ",
  $session->status_code() . ":" . $session->status_detail());
print "Removed DHCP exclusion range template from DHCP Range successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::RangeTemplate, Infoblox::DHCP::NetworkTemplate

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::ExpandNetwork - Join or Expand an existing network

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
- METHODS
  - auto_create_reversezone( )
  - network( )
  - prefix( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::ExpandNetwork - Join or Expand an existing network

DESCRIPTION

Expanding/Joining multiple networks into a larger network is the opposite of spliting a network. You can select a network and expand it into a larger network with smaller netmask.

CONSTRUCTOR

```
#Constructor a DHCP Expand Network object
my $expand_network = Infoblox::DHCP::ExpandNetwork->new (
       network                 => $Network|$IPv6Network, #Required
       prefix                  => $num,                 #Required
       auto_create_reversezone => "true" | "false",      #Optional / Default is "false"
    );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a DHCP ExpandNetwork object.

Infoblox::Session->add( )

Use this method to expand an existing network on the Infoblox appliance. See Infoblox::Session->add( ) for parameters and return values

Example

```
# construct the network to be expanded
my $network = Infoblox::DHCP::Network->new(
         network => "123.0.0.0/16",
           );

# construct a Expand Network object
my $expand_network = Infoblox::DHCP::ExpandNetwork->new (
        network                 => $network,
        prefix                  => "13",
        auto_create_reversezone => "false"
);

# Submit the Expand network operation.
$session->add($expand_network);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute value of a Expand Network.

auto_create_reversezone( )

Use this method to set the auto_create_reversezone flag. When set, this flag will automatically create reverse-mapping zones for the subnet.

Include the specified parameter to set the attribute value.

Parameter

Specify "true" to set auto_create_reversezone flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Set auto_create_reversezone
$split_network->auto_create_reversezone("true");
```

network( )

Use this method to specify the network to be expanded.

Include the specified parameter to set the attribute value.

Parameter

The valid values are either Infoblox::DHCP::Network or Infoblox::DHCP::IPv6Network objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Specify network
my $network = Infoblox::DHCP::Network->new(
                    network => "123.0.0.0/16",
                    );
$expand_network->network($network);

#Specify IPv6 network
my $network = Infoblox::DHCP::IPv6Network->new(
                    network => "12::/68",
                    );
$expand_network->network($network);
```

prefix( )

Use this method to specify the appropriate subnet mask to be expanded on the network.

Include the specified parameter to set the attribute value.

Parameter

Number with the subnet mask value. This value must be 1 smaller then the subnet mask of the network to be expanded. The minimuim value is 8.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Specify prefix
$expand_network->prefix(13);
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, modify, and remove an object. This sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
```

```
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create network prior to splitting the network my $network = Infoblox::DHCP::Network->new(network => "123.0.0.0/16", comment => "add network",); unless($network){ die("Construct network object fail: ", Infoblox::status_code(). ":" .Infoblox::status_detail()); }

```
print"Network object created successfully.\n";

#Verify if the network exists
my $object = $session->get(
               object  => "Infoblox::DHCP::Network",
               network => "123.0.0.0/16"
               );
unless($object){
       print"Network does not exist on server, safe to add the network.\n";
       $session->add($network)
               or die("Add network failed: ",
                       $session->status_code(). ":" .$session->status_detail());
       }

print"Network added successfully.\n";
```

#Expand the network

```
my $expand_network = Infoblox::DHCP::ExpandNetwork->new (
                       network                 => $network,
                       prefix                  => "13",
                       auto_create_reversezone => "false",
                     );
unless($expand_network){
       die("Construct expand network object fail: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
       }

print"Expand network object created successfully.\n";

#Expand the network
$session->add( $expand_network )
         or die("Expand network fail:",
                    $session->status_code(). ":" .$session->status_detail());

print "Expand network successfully.\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::DHCP::Network

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::FailOver - DHCP Failover object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - primary( )
  - secondary( )
  - comment( )
  - extensible_attributes( )
  - load_balance_split( )
  - max_load_balance_delay( )
  - max_client_lead_time( )
  - max_unacked_updates( )
  - max_response_delay( )
  - primary_state( )
  - recycle_leases( )
  - secondary_state( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::FailOver - DHCP Failover object.

DESCRIPTION

DHCP failover is a protocol designed to allow a backup DHCP server to take over for a main server if the main server is taken off the network for any reason. DHCP failover can be used to configure two DHCP servers to operate as a redundant pair.

CONSTRUCTOR

```
 my $dhcp_failover = Infoblox::DHCP::FailOver->new(
     name                  => $string,                          #Required
     primary               => $ipv4addr | $Member               #Required
     secondary             => $ipv4addr | $Member               #Required
     comment               => $string,                          #Optional / Default is empty
     extensible_attributes  => { $string => $string | $num, $string => [ $string | $num, ... ], ... },
#Optional / Default is undefined
     load_balance_split     => $num,                            #Optional / Default value is 128
     max_load_balance_delay => $num,                            #Optional / Default value is 3
     max_client_lead_time   => $num,                            #Optional / Default value is 3600
     max_unacked_updates    => $num,                            #Optional / Default value is 10
     max_response_delay     => $num,                            #Optional / Default value is 60
     recycle_leases         => "true" | "false" | undef,         #Optional / Default value is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that can be applied to a DHCP failover object.

Infoblox::Session->add( )

Use this method to add a DHCP failover object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Create a DHCP Member object
my $memberdhcp   Infoblox::DHCP::Member   new
```

```
                =                        ->    (
      name     => "infoblox.localdomain",
      ipv4addr => "192.168.1.2"
);

#Construct an object
my $dhcp_failover = Infoblox::DHCP::FailOver->new(
    name                  => "peer1",
    primary               => $memberdhcp,
    secondary             => "1.1.1.1",
    load_balance_split    => 128,
    max_load_balance_delay => 3,
    max_client_lead_time  => 3600,
    max_unacked_updates   => 10,
    max_response_delay    => 60,
    recycle_leases        => "false",
);

# Submit for addition
my $response = $session->add( $dhcp_failover );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching DHCP failover objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

Apply the following attributes to get a specific DHCP failover object:

name - Required. Name of a DHCP failover in string format.

Example

```
my @retrieved_objs = $session->get(
                     object     => "Infoblox::DHCP::FailOver",
                     name       => "peer1" );
```

Infoblox::Session->modify( )

Use this method to modify a DHCP failover object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the load_balance_split.
$dhcp_failover->load_balance_split(130);
# Submit modification
my $response = $session->modify( $dhcp_failover );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP failover object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP failover object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object  => "Infoblox::DHCP::FailOver",
    name    => "peer1" );
# Find the desired object from the retrieved list.
my $desired_dhcp_failover = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_dhcp_failover );
```

Infoblox::Session->search( )

Use this method to search for DHCP failover objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

Apply the following attributes, to search for a DHCP failover object:

```
name    - Optional. The name of a DHCP failover object. Regular expression supported.
comment - Optional. The comment of the DHCP failover object. Regular expression supported.
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for DHCP failover object with name as peer1
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::FailOver",
    name    => "pe.*" );

# search all DHCP failover associations with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DHCP::FailOver",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP failover object.

name( )

Use this method to set or retrieve the name of a DHCP failover object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $dhcp_failover->name();
#Modify name
$dhcp_failover->name("peer2");
```

primary( )

Use this method to set or retrieve primary server of a DHCP failover object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify external primary server in IPv4 address format or grid primary server in a defined Infoblox::DHCP::Member object. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get primary
my $primary = $dhcp_failover->primary();
#Modify primary
$dhcp_failover->primary($member); #$member is a Infoblox::DHCP::Member object
```

secondary( )

Use this method to set or retrieve secondary server of a DHCP failover object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify external secondary server in IPv4 address format or grid secondary server in a defined Infoblox::DHCP::Member object.

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get secondary
my $secondary = $dhcp_failover->secondary();
#Modify secondary
$dhcp_failover->secondary("2.2.2.2");
```

comment( )

Use this method to set or retrieve a descriptive comment about a DHCP failover object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $dhcp_failover->comment();
# Modify comment
$dhcp_failover->comment("add a network");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP failover object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $dhcp_failover->extensible_attributes();
#Modify extensible attributes
$dhcp_failover->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter',
'Tom' ] });
```

load_balance_split( )

Use this method to set or retrieve a load balancing split value of a DHCP failover object.

Load balancing split determines which server handles IP address requests. This parameter is specified on the primary only.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the value of the maximum load balancing delay in a 8-bit integer format (range from 0 to 255). The default value is 128.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get load_balance_split
my $load_balance_split = $dhcp_failover->load_balance_split();
#Modify load_balance_split
$dhcp_failover->load_balance_split(200);
```

max_load_balance_delay( )

Use this method to set or retrieve the maximum load balancing delay value of a DHCP failover object.

Maximum load balancing delay specifies the cutoff after which load balancing is disabled. The cutoff is based on the number of seconds since the client sent its first DHCPDISCOVER or DHCPREQUEST message. For instance, if one of the failover peers gets into a state where it is responding to failover messages, but not responding to some client requests, the other failover peer will take over its client load automatically as the clients retry.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the value of the maximum load balancing delay in a 32-bit integer format (range from 0 to 4294967295) that represents the duration in seconds. The default value is 3.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get max_load_balance_delay
my $max_load_balance_delay = $dhcp_failover->max_load_balance_delay();
#Modify max_load_balance_delay
$dhcp_failover->max_load_balance_delay("500");
```

max_client_lead_time( )

Use this method to set or retrieve the maximum client lead time value of a DHCP failover object.

Maximum client lead time value specifies the length of time that a failover peer can renew a lease without contacting the other peer. The larger the number, the longer it takes for the peer server to recover IP addresses after moving to the Partner Down mode. The smaller the number, the more load your servers experience when they are not communicating. This is specified on the primary server only.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the value of the maximum client lead time in a 32-bit integer format (range from 0 to 4294967295) that represents the duration in seconds. The default value is 3600.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get max_client_lead_time
my $max_client_lead_time = $dhcp_failover->max_client_lead_time();
#Modify max_client_lead_time
$dhcp_failover->max_client_lead_time(600);
```

max_unacked_updates( )

Use this method to set or retrieve the maximum number of unacked updates value of a DHCP failover object.

Maximum number of unacked updates value specifies how many update messages the server can send before it should receive an ACK from the failover peer. If no ACK is received after these messages are sent, failover occurs.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the value of the maximum number of unacked updates in a 32-bit integer format (range from 0 to 4294967295) that represents the number of messages. The default value is 10.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get max_unacked_updates
my $max_unacked_updates = $dhcp_failover->max_unacked_updates();
#Modify max_unacked_updates
$dhcp_failover->max_unacked_updates(700);
```

max_response_delay( )

Use this method to set or retrieve the maximum response delay value of a DHCP failover object.

Maximum response delay value before failover specifies how many seconds can transpire before the primary server assumes its peer (the secondary server) is not sending messages due to failure.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the value of the maximum response delay in a 32-bit integer format (range from 0 to 4294967295) that represents the duration in seconds. The default value is 60.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get max_response_delay
my $max_response_delay = $dhcp_failover->max_response_delay();
#Modify max_response_delay
$dhcp_failover->max_response_delay(800);
```

primary_state( )

Use this method to retrieve the primary server status of a DHCP failover object.

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get primary_state
my $primary_state = $dhcp_failover->primary_state();
```

recycle_leases( )

Use this method to set or retrieve the recycle leases flag of a DHCP failover object.

If the flag is enabled, the leases are kept in recycle bin until one week after expiration and when the flag is disabled, the leases are irrecoverably deleted.

Parameter

Specify "true" to set the recycle_leases flag or "false" to deactivate/unset it. The default value for this field is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $dhcp_failover->recycle_leases();
#Modify recycle_leases
$dhcp_failover->recycle_leases("true");
#Un-override recycle_leases
$dhcp_failover->recycle_leases(undef);
```

secondary_state( )

Use this method to retrieve the secondary server status of a DHCP failover object.

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get secondary_state
my $secondary_state = $dhcp_failover->secondary_state();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP failover object such as add, search, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP failover object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a DHCP Member object
my $memberdhcp = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2"
);
unless($memberdhcp) {
       die("Construct member failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Member object created successfully\n";

#Create the Network object with member
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    comment => "add network",
    members => [ $memberdhcp ]
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
    or die("Add Network object failed: ",
```

```
            $session->status_code() . ":" . $session->status_detail());
print "Network object added to Infoblox appliance successfully\n";
```

#Create a DHCP failover object

```
my $dhcp_failover = Infoblox::DHCP::FailOver->new(
    name                  => "peer1",
    primary               => $memberdhcp,
    secondary             => "1.1.1.1",
    load_balance_split     => 128,
    max_load_balance_delay => 3,
    max_client_lead_time   => 3600,
    max_unacked_updates    => 10,
    max_response_delay     => 60,
    recycle_leases         => "true",
);
unless($dhcp_failover) {
       die("Construct DHCP failover object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP failover object created successfully\n";

#Add a DHCP failover object into the Infoblox appliance through a session
$session->add($dhcp_failover)
    or die("Add DHCP failover object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP failover object added to the network successfully\n";
```

#Get and modify a DHCP failover

```
#Get a DHCP failover through the session
my @retrieved_objs = $session->get(
    object  => "Infoblox::DHCP::FailOver",
    name    => "peer1",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP failover failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP failover object found at least 1 matching entry\n";

#Modify one of the attributes of the specified DHCP failover
$object->max_client_lead_time(3500);

#Apply the changes
$session->modify($object)
    or die("Modify DHCP failover failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP failover object modified successfully \n";
```

#Search a DHCP failover

```
# search for DHCP failover object with name starting with letter "pe"
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::FailOver",
    name   => "pe.*"
);

my $object = $retrieved_objs[0];
unless ($object) {
    die("Search DHCP failover failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DHCP failover object found at least 1 matching entry\n";
```

#Add a DHCP range with failover

```
#Construct the DHCP range with failover peer1
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr       => "10.0.0.10",
        network        => "10.0.0.0/8",
        start_addr     => "10.0.0.1",
        failover_assoc => "peer1"
);
unless($dhcp_range) {
       die("Construct DHCP Range object with failover failed: ",
```

```
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object with failover peer1 created successfully\n";

#Add the DHCP Range object with failover into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object with failover failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object with failover added to the network successfully\n";

#submit the network object for removal
$session->remove($network)
    or die("Remove Network failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network removed successfully \n";
```

#Remove a DHCP failover

```
#Get a DHCP failover through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FailOver",
    name   => "peer1"
);

my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP failover failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP failover object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove DHCP failover failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP failover object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify(), Infoblox::DHCP::Member

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Filter::MAC - DHCP MAC Address Filter object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - default_mac_address_expiration( )
  - enforce_expiration_times( )
  - extensible_attributes( )
  - name( )
  - reserved_for_infoblox( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Filter::MAC - DHCP MAC Address Filter object.

DESCRIPTION

An Infoblox appliance can filter address requests by the MAC address and/or vendor prefix (i.e., the first 6 hexadecimal characters in MAC address) of a requesting host. The filter instructs the infoblox appliance either to grant or deny an address request if the requesting host matches the filter.

CONSTRUCTOR

```
my $mac_filter = Infoblox::DHCP::Filter::MAC->new(
      name                          => $string,          #Required
      comment                       => $string,          #Optional / Default is empty
      default_mac_address_expiration => $num,            #Optional / Default is 0 for never expire
      enforce_expiration_times      => "true" | "false", #Optional / Default is "true"
      extensible_attributes         => { $string => $string | $num, $string => [ $string | $num, ... ],
... }, #Optional / Default is undefined
      reserved_for_infoblox         => $string,          #Optional / Default is empty
);
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that can be applied to a DHCP MAC Address Filter object.

Infoblox::Session->add( )

Use this method to add a DHCP MAC Address Filter object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a DHCP MAC Address Filter object
my $mac_filter = Infoblox::DHCP::Filter::MAC->new (
      name                          => "filter_script",
      comment                       => "Created through script",
      default_mac_address_expiration => "70",
      enforce_expiration_times      => "true",
      reserved_for_infoblox         => "This is reserved for infoblox",
);
#Submit for addition
my $response = $session->add($mac_filter)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP MAC Address Filter object:

 name - Required. Name of the MAC Address Filter.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
#Get DHCP MAC Address Filter object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::MAC",
    name   => "filter_script",
);

# get all objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object                => "Infoblox::DHCP::Filter::MAC",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP MAC Address Filter object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the default_mac_address_expiration
$mac_filter->default_mac_address_expiration("1234");
#Submit modification
my $response = $session->modify( $mac_filter );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP MAC Address Filter object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP MAC Address Filter object, then submit this object for removal.

Example

```
#Get DHCP MAC Address Filter object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::MAC",
    name   => "filter_script",
);
#Find the desired object from the retrieved list.
my $desired_mac_filter = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_mac_filter );
```

Infoblox::Session->search( )

Use this method to search for DHCP MAC Address Filter objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP MAC Address Filter object:

 name - Name of the MAC Address Filter (regular expression).
 comment - Comment string for the MAC Address Filter (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.

Either name or comment must be specified.
```

Example

```
# search for all DHCP MAC Address Filter objects whose name starts with "filter".
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::Filter::MAC",
    name    => "filter.*",
);
```

```
# search for all DHCP MAC Address Filter objects whose comment starts with "user1".
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::Filter::MAC",
    comment    => "user1.*",
);

# search all DHCP MAC Address Filter associations with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DHCP::Filter::MAC",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that can be used to set and retrieve the attribute values of a DHCP MAC Address Filter object.

comment( )

Use this method to set or retrieve a descriptive comment of a DHCP MAC Address Filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $mac_filter->comment();
#Modify comment
$mac_filter->comment("Modified DHCP MAC Address Filter comment");
```

default_mac_address_expiration( )

Use this method to set or retrieve the default mac address expiration time of a DHCP MAC Address Filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

By default, the MAC address filter never expires; otherwise, it is the absolute interval when the MAC address filter expires. The maximum value can extend up to 4294967295 secs. The minimum value is 60 secs (1 min).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get default_mac_address_expiration
my $default_mac_address_expiration = $mac_filter->default_mac_address_expiration();
#Modify default_mac_address_expiration
$mac_filter->default_mac_address_expiration("1234");
```

enforce_expiration_times( )

Use this method to set or retrieve the enforce_expiration_times flag of a DHCP MAC Address Filter object.

Determines whether lease requests will be denied from expired MAC Addresses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the expired flag or "false" to deactivate/unset it. Default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enforce_expiration_times
my $enforce_expiration_times = $mac_filter->enforce_expiration_times();
#Modify enforce_expiration_times
$mac_filter->enforce_expiration_times("false");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP MAC Address Filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $mac_filter->extensible_attributes();
#Modify extensible attributes
$mac_filter->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

name( )

Use this method to set or retrieve the name of a DHCP MAC Address Filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the DHCP MAC Address filter. Maximum length up to 1024 bytes is supported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $mac_filter->name();
#Modify name
$mac_filter->name("filter1");
```

reserved_for_infoblox( )

Set this flag to reserve the MAC address filter for an Infoblox appliance.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Reserved to write comment related to this particular MAC address filter. The length of comment cannot exceed 1024 bytes long.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get reserved_for_infoblox
```

```
my $reserved_for_infoblox = $mac_filter->reserved_for_infoblox();
#Modify reserved_for_infoblox
$mac_filter->reserved_for_infoblox("reserved for Infoblox");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP MAC Address Filter object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP MAC Address Filter object

```
#Create a DHCP MAC Address Filter object
my $mac_filter = Infoblox::DHCP::Filter::MAC->new (
       name                          => "script_filter",
       comment                       => "Create through script",
       default_mac_address_expiration => "70",
       enforce_expiration_times       => "true",
       reserved_for_infoblox          => "This is reserved for infoblox",
);

unless($mac_filter) {
     die("Construct mac address filter failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "MAC Address Filter object created successfully\n";

#Verify if the DHCP MAC Address Filter exists in the Infoblox appliance
my $object = $session->get(object => "Infoblox::DHCP::Filter::MAC", name => "script_filter");
unless ($object) {
   print "Mac address filter does not exist on server, safe to add the mac address filter\n";
   $session->add($mac_filter)
      or die("Add mac address filter failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "DHCP MAC Address Filter added successfully\n";
```

#Search for a specific DHCP MAC Address Filter object

```
#Search for all DHCP MAC address filter objects
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::Filter::MAC",
    name   => ".*" );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search for MAC Address Filter object failed: ",
              $session->status_code() . ":" . $session->status_detail());
}
print "Search MAC Address Filter object found at least 1 matching entry\n";
```

#Get and modify a MAC Address Filter object

```
#Get MAC Address Filter object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::MAC",
    name   => "script_filter" );
```

```
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get MAC Address Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get MAC Address Filter object found at least 1 matching entry\n";

#Modify one of the attributes of the obtained MAC Address Filter object
$object->default_mac_address_expiration("1234");

#Apply the changes
$session->modify($object)
    or die("Modify MAC Address Filter object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "MAC Address Filter object modified successfully \n";
```

#Remove a MAC Address Filter object

```
#Get MAC Address Filter object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::MAC",
    name   => "script_filter",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get MAC Address Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get MAC Address Filter object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove MAC Address Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "MAC Address Filter object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::MAC, Infoblox::DHCP::Filter::RelayAgent, Infoblox::DHCP::Lease, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->modify(), Infoblox::Session->remove()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Filter::MatchRule - DHCP Option Filter MatchRule object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - filter( )
  - name( )
  - comment( )
  - num( )
  - is_substring( )
  - reserved_for_infoblox( )
  - substring_offset( )
  - substring_length( )
  - value( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Filter::MatchRule - DHCP Option Filter MatchRule object.

DESCRIPTION

A MatchRule defines the (option, value) pair match rule for an option filter. In the ISC DHCP terms, it is the matched if statement within the class declaration. To define a match rule, add MatchRule to the DHCP Filter object.

CONSTRUCTOR

```
  my $match_rule = Infoblox::DHCP::Filter::MatchRule->new (
          filter                => $string,                          #Required,
          value                 => $num | $string |"true"|"false", #Required
          name                  => $string,                          #Optional - One of name or num must be
specified / Default is undefined
          comment               => $string,                          #Optional / Default is empty
          num                   => $num,                             #Optional - One of name or num must be
specified / Default is undefined
          is_substring          => "true" | "false",                 #Optional / Default is "false"
          reserved_for_infoblox => $string,                          #Optional / Default is empty
          substring_offset      => $string,                          #Optional / Default is empty
          substring_length      => $string,                          #Optional / Default is empty
   );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that can be applied to a DHCP Option Filter MatchRule object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $match_rule = Infoblox::DHCP::Filter::MatchRule->new (
filter => "filter1",
name   => "vendor-class-identifier",
value  => "solaris-i86pc"
);

# Submit for addition
```

```
my $response = $session->add( $match_rule );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP Filter MatchRule object:

filter   - Required. A filter name in string format.
name/num - Required. A match option name or code in string format.
value    - Required. A match option value in string format.
```

Example

```
my @retrieved_objs = $session->get(
object => "Infoblox::DHCP::Filter::MatchRule",
filter => "filter1",
name   => "vendor-class-identifier",
value  => "solaris-i86pc");
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the value.
$match_rule->value("solaris");
# Submit modification
my $response = $session->modify( $match_rule );
```

Infoblox::Session->remove( )

Use this method to submit to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
object => "Infoblox::DHCP::Filter::MatchRule",
filter => "filter1",
num    => "60",
value  => "solaris");

# Find the desired object from the retrieved list.
my $desired_match_rule = $retrieved_objs[0];

# Submit for removal
my $response = $session->remove( $desired_match_rule );
```

Infoblox::Session->search( )

Use this method to search for DHCP Filter::MatchRule objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP Filter MatchRule object:

filter   - Required. A filter name in string format (regular expression).
name/num - Optional. A match option name or code in string format (regular expression if name).
value    - Optional. A match option value in string format.
comment  - Optional . A comment in string format (regular expression).
```

Example

```
# search for all DHCP Filter::MatchRule objects.
my @retrieved_objs = $session->search(
 object => "Infoblox::DHCP::Filter::MatchRule",
 filter => "f*"
);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of an Option Filter MatchRule.

filter( )

Use this method to set or retrieve filter name of a match rule.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired filter name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get filter
my $filter = $match_rule->filter();
#Modify filter
$match_rule->filter("filter2");
```

name( )

Use this method to set or retrieve the match option name of a match rule.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value. parameter to retrieve the attribute value.

Parameter

Desired match option name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $match_name = $match_rule->name();
#Modify name
$match_rule->name(dhcp-client-identifier);
```

comment( )

Use this method to set or retrieve the descriptive comment of a DHCP matchrule filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $match_rule->comment();
#Modify comment
$match_rule->comment("Modified DHCP matchrule filter object comment");
```

num( )

Use this method to set or retrieve the match option code of a match rule.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired match option code in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get num
my $match_num = $match_rule->num();
#Modify num
$match_rule->num("77");
```

is_substring( )

Use this method to set or retrieve the is_substring value of a match rule.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Boolean values "true" or "false". The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value

Example

```
#Get is_substring
my $is_substring = $match_rule->is_substring();
#Modify is_substring
$match_rule->is_substring("true");
```

reserved_for_infoblox( )

Use this method to set or retrieve the reserved_for_infoblox value of a match rule.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Reserved to write comment related to this particular option match rule filter. The length of comment cannot exceed 1024 bytes long.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value

Example

```
#Get reserved_for_infoblox
my $reserved_for_infoblox = $match_rule->reserved_for_infoblox();
#Modify reserved_for_infoblox
$match_rule->reserved_for_infoblox("reserved for Infoblox");
```

substring_offset( )

Use this method to set or retrieve the substring_offset value of a match rule.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired substring offset number in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get substring_offset
my $offset = $match_rule->substring_offset();
#Modify substring_offset
$match_rule->substring_offset("2");
```

substring_length( )

Use this method to set or retrieve the substring_length value of a match rule.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired substring length number in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get substring_length
my $substring_length = $match_rule->substring_length();
#Modify substring_length
$match_rule->substring_length("9");
```

value( )

Use this method to set or retrieve the value of the DHCP filter rule.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the value of the DHCP filter rule based on the name of the DHCP option. Valid values are

```
  "boolean"                 - Specify "true" or "false" for boolean values.

  "ipv4addr"                - Specify IPV4 address. An IPv4 address is a 32-bit number in dotted
decimal notation.
                              It consists of four 8-bit groups of decimal digits separated
                              by decimal points (example: 192. 168.1.2).

  "ipv4addrs"               - Specify the value in array reference that contains IPv4 addresses.

  "string"                  - Specify the value in ASCII text string or
                              a list of hexadecimal characters (0-9, a-f, or A-F) without spaces and
separated by colons.

  "text"                    - Specify the value in ASCII text string.

  "8-bit unsigned integer"  - Specify the numeric value between 0 and 255.

  "16-bit unsigned integer" - Specify the numeric value between 0 and 65535.

  "32-bit unsigned integer" - Specify the numeric value between 0 and 4294967295.

  "8-bit signed integer"    - Specify the numeric value between -128 and 127.

  "16-bit signed integer"   - Specify the numeric value between -32768 and 32767.

  "32-bit signed integer"   - Specify the numeric value between -2147483648 and 2147483647.
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get value
my $value = $match_rule->value();
#Modify value
$match_rule->value("solaris-i86pc");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, get, modify, search and remove. Also, this sample also includes error handling for the operations.

#Preparation prior to a DHCP Filter::MatchRule insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox Appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);

unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";

  my $opt1 = Infoblox::DHCP::Option->new(
      name  => "user-class" ,
      value => "77"
   );
```

#Create the Option Filter object

```
my $option_filter = Infoblox::DHCP::Filter::Option->new(
      name        => 'filter1' ,
      option_list => [$opt1],

);
unless($option_filter) {
       die("Construct option Filter object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Option Filter object created successfully\n";

#Add the filter rule option object into the Infoblox appliance>

$session->add($option_filter)
    or die("Add option Filter object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Option Filter object added to Infoblox appliance successfully\n";
```

#Create a DHCP Filter::MatchRule

```
#Construct a DHCP Filter::MatchRule object

my $match_rule = Infoblox::DHCP::Filter::MatchRule->new(
    filter     => "filter1",
    name       => "subnet-mask",
    value      => "255.0.0.0",
);
unless ($match_rule) {
   die("Construct DHCP Filter::MatchRule failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

print "DHCP Filter::MatchRule object created successfully\n";

# Add the DHCP Filter::MatchRule object to Infoblox DHCPone appliance through a session

$session->add($match_rule)
    or die("Add Filter::MatchRule failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "DHCP Filter::MatchRule object added to server successfully\n";
```

#Get and modify a DHCP Filter::MatchRule

```
#Get Filter::MatchRule through the session
```

```
my @retrieved_objs = $session->get(
    object  => "Infoblox::DHCP::Filter::MatchRule",
    filter  => "filter1",
    name    => "subnet-mask",
    value   => "255.0.0.0",
);

my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Filter::MatchRule failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Get DHCP Filter::MatchRule object found at least 1 matching entry\n";

#Modify one of the attributes of the specified Filter::MatchRule

$object->value("255.255.0.0");
#Apply the changes
$session->modify($object)
    or die("Modify Filter::MatchRule failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DHCP Filter::MatchRule object modified successfully \n";
```

#Search for match rule filter

```
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::Filter::MatchRule",
    name   => "subnet-mask"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search MatchRule filter failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search MatchRule Filter object found at least 1 matching entry\n";
```

#Remove a DHCP Filter::MatchRule

```
#Get Filter::MatchRule through the session

my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::MatchRule",
    filter => "filter1",
    name   => "subnet-mask",
    value  => "255.255.0.0",
);

my $object = $retrieved_objs[0];

unless ($object) {
    die("Get Filter::MatchRule failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Get DHCP Filter::MatchRule object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove Filter::MatchRule failed: ",
        $session->status_code() . ":" . $session->status_detail());

print "DHCP Filter::MatchRule object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Filter::NAC - DHCP NAC Filter object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - expression( )
  - name( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Filter::NAC - DHCP NAC Filter object.

DESCRIPTION

If NAC authentication is configured, the appliance receives authentication responses from NAC authentication servers, and it grants or denies a lease request if the authentication response matches conditions defined by the NAC filters.

CONSTRUCTOR

```
my $nac_filter = Infoblox::DHCP::Filter::NAC->new(
      name                          => $string,          #Required
      expression                    => $string,          #Optional / Default is empty
      comment                       => $string,          #Optional / Default is empty
);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that can be applied to a DHCP NAC Filter object.

Infoblox::Session->add( )

Use this method to add a DHCP NAC Filter object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a DHCP NAC Filter object
my $nac_filter = Infoblox::DHCP::Filter::NAC->new (
      name                          => "filter_script",
      comment                       => "Created through script",
      expression                    => 'Sophos.ComplianceState="PartialCompliant"',
);
#Submit for addition
my $response = $session->add($nac_filter)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP NAC Filter object:

 name - Required. Name of the NAC Filter.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
#Get DHCP NAC Filter object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::NAC",
    name   => "filter_script",
);

# get all objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object                => "Infoblox::DHCP::Filter::NAC",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP NAC Filter object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the default_nac_address_expiration
$nac_filter->comment("modified comment");
#Submit modification
my $response = $session->modify( $nac_filter );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP NAC Filter object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP NAC Filter object, then submit this object for removal.

Example

```
#Get DHCP NAC Filter object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::NAC",
    name   => "filter_script",
);
#Find the desired object from the retrieved list.
my $desired_nac_filter = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_nac_filter );
```

Infoblox::Session->search( )

Use this method to search for DHCP NAC Filter objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP NAC Filter object:

 name - Name of the NAC filter (regular expression).
 comment - Comment string for the NAC filter (regular expression).
 extensible_attributes - Optional. A hash reference that contains extensible attributes.

Either a name or comment must be specified.
```

Example

```
# search for all DHCP NAC Filter objects whose name starts with "filter".
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::Filter::NAC",
    name    => "filter.*",
);

# search for all DHCP NAC Filter objects whose comment starts with "user1".
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::Filter::NAC",
    comment    => "user1.*",
);

# search all DHCP NAC Filter associations with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DHCP::Filter::NAC",
```

```
extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that can be used to set and retrieve the attribute values of a DHCP NAC Filter object.

comment( )

Use this method to set or retrieve a descriptive comment of a DHCP NAC Filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format, with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $nac_filter->comment();
#Modify comment
$nac_filter->comment("Modified DHCP NAC Filter comment");
```

expression( )

Use this method to set or retrieve the conditional expression of a DHCP NAC Filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An expression in string format, with a maximum of 4096 characters. The expression must comply with the NAC filter expression syntax.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting expression
my $expression = $nac_filter->expression( );
#Modifying expression
$nac_filter->expression('Sophos.ComplianceState="PartialCompliant"');
```

name( )

Use this method to set or retrieve the name of a DHCP NAC Filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the DHCP NAC filter. The maximum length supported is 1024 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $nac_filter->name();
#Modify name
$nac_filter->name("filter1");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP NAC Filter object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP NAC Filter object

```
#Create a DHCP NAC Filter object
my $nac_filter = Infoblox::DHCP::Filter::NAC->new (
       name                          => "script_filter",
       comment                       => "Create through script",
       expression                     => 'Sophos.ComplianceState="PartialCompliant"',
);

unless($nac_filter) {
     die("Construct nac filter failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "NAC Filter object created successfully\n";

#Verify if the DHCP NAC Filter exists in the Infoblox appliance
my $object = $session->get(object => "Infoblox::DHCP::Filter::NAC", name => "script_filter");
unless ($object) {
   print "nac filter does not exist on server, safe to add the nac filter\n";
   $session->add($nac_filter)
      or die("Add nac filter failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "DHCP NAC Filter added successfully\n";
```

#Search for a specific DHCP NAC Filter object

```
#Search for all DHCP NAC filter objects
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::Filter::NAC",
    name   => ".*" );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search for NAC Filter object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search NAC Filter object found at least 1 matching entry\n";
```

#Get and modify a NAC Filter object

```
#Get a NAC Filter object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::NAC",
    name   => "script_filter" );

my $object = $retrieved_objs[0];
unless ($object) {
    die("Get NAC Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get NAC Filter object found at least 1 matching entry\n";
```

```
#Modify one of the attributes of the obtained NAC Filter object
$object->comment("Modified comment");

#Apply the changes
$session->modify($object)
    or die("Modify NAC Filter object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "NAC Filter object modified successfully \n";
```

#Remove a NAC Filter object

```
#Get a NAC Filter object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::NAC",
    name   => "script_filter",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get NAC Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get NAC Filter object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove NAC Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "NAC Filter object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::Filter::RelayAgent, Infoblox::DHCP::Lease, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->modify(), Infoblox::Session->remove()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Filter::Option - DHCP filter option object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - boot_file( )
  - boot_server( )
  - comment( )
  - extensible_attributes( )
  - lease_time( )
  - next_server( )
  - option_list( )
  - option_space( )
  - pxe_lease_time( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Filter::Option - DHCP filter option object

DESCRIPTION

A Filter::Option defines a DHCP option filter. In the ISC DHCP terms, it defines a class of clients that match a particular (option, value) pair. To define an option filter, add Option to the DHCP Filter object.

CONSTRUCTOR

```
  my $optionFilter = Infoblox::DHCP::Filter::Option->new(
                name                  => $string,                          #Required
                comment               => $string,                          #Optional / Default is empty
                boot_file             => $filename | undef,                 #Optional / Default is
undefined
                boot_server           => $fqdn | undef,                     #Optional / Default is
undefined
                extensible_attributes  => { $string => $string | $num, $string => [ $string | $num, ...
], ... }, #Optional / Default is undefined
                lease_time            => $time,                            #Optional / Default is empty
                next_server           => $fqdn | undef,                     #Optional / Default is
undefined
                option_list           => [$Option1, $Option2,...],          #Optional / Default is empty
                option_space          => $string,                          #Optional / Default is "DHCP"
                pxe_lease_time        => $time,                            #Optional / Default is empty
               );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that can be applied to a DHCP filter option object.

Infoblox::Session->add( )

Use this method to add a DHCP filter option object to the Infoblox device. See Infoblox::Session->add() for parameters and return values.

Example

```
      #Construct an object
      my  $optionFilter = Infoblox::DHCP::Filter::Option->new(
            name             => "option_filter",
```

```
        );

# Submit for adding a option Filter
my $response = $session->add( $optionFilter );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching DHCP option filter objects from the Infoblox device. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP relayagent filter object:

name - Required. An option filter name in string format.
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my  @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::Option" ,
    name   => "option_filter",
);

# get all objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object                => "Infoblox::DHCP::Filter::Option",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP option filter object in the Infoblox device. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify boot_server method.
$object->boot_server("myboot_server.com");
# Submit modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP option filter object from the Infoblox device. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my @retrieved_objs = $session->get(
          object  => "Infoblox::DHCP::Filter::Option" ,
          "name"  => "option_filter",
       );

# find the desired object from retrieved list.
my $desired_obj = $retrieved_objs[0];

# Submit for removal
my $response = $session->remove( $desired_obj);
```

Infoblox::Session->search( )

Use this method to search for DHCP option filter object in the Infoblox device. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP Network object:

 name - Required. An option filter name in string format (regular expression).
 comment - Optional . A comment in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DHCP option filter objects
     my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::Filter::Option",
    name   => "option.*",
);

# search all DHCP option filter associations with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DHCP::Filter::Option",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of DHCP option filter object

name( )

Use this method to set or retrieve the name of a DHCP option filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the name in string format. The default value for this field is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $name = $optionFilter->name();
#Modify name
$optionFilter->name("option_filter");
```

boot_file( )

Use this method to set or retrieve a boot_file attribute of a DHCP filter option object. If this attribute is not set, it will inherit the member level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the file that the client must download.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get boot_file
my $boot_file = $optionFilter->boot_file();
# Modify boot_file
$optionFilter->boot_file("bootfile1");
#Un-override boot_file
$optionFilter->boot_file(undef);
```

boot_server( )

Use this method to set or retrieve a bootserver attribute of a DHCP filter option object. You can specify the name and/or IP address of the boot server that host needs to boot. If this attribute is not set, it will inherit the member level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get boot_server
my $bootserver = $optionFilter->bootserver();
# Modify boot_server
$optionFilter->boot_server("abc.domain.com");
#Un-override boot_server
$optionFilter->boot_server(undef);
```

comment( )

Use this method to set or retrieve the descriptive comment of a DHCP filter option object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $optionFilter->comment();
#Modify comment
$optionFilter->comment("Modified DHCP filter option object comment");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP Filter Option object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $optionFilter->extensible_attributes();
#Modify extensible attributes
$optionFilter->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom'
] });
```

lease_time( )

Use this method to set or retrieve a lease_time attribute of a DHCP filter option object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter appropriate values in seconds.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get lease_time
my $lease_time = $optionFilter->lease_time();
# Modify lease_time
$optionFilter->lease_time("7200");
```

next_server( )

Use this method to set or retrieve a next_server attribute of a DHCP filter option object. You can specify the name and/or IP address of the next server that the host needs to boot. If this attribute is not set, it will inherit the member level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). The next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get next_server
my $nextserver = $optionFilter->next_server();
# Modify next_server
$optionFilter->next_server("blue.domain.com");
#Un-override next_server
$optionFilter->next_server(undef);
```

option_list( )

Use this method to set or retrieve the option_list of a DHCP filter option object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option objects. Option list describe filter option configuration settings and various services.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get option_list
my $options = $optionFilter->option_list();
#Modify option_list
$optionFilter->option_list([$option1]);
```

option_space( )

Use this method to retrieve the option_space of a DHCP filter option object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the option space value in string format. The default value of option_space is DHCP.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get option_space
my $options = $optionFilter->option_space();
#Modify option_space
$optionFilter->option_space("Infoblox_DHCP");
```

pxe_lease_time( )

Use this method to set or retrieve a pxe_lease_time attribute of a DHCP filter option object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter appropriate values in seconds.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get pxe_lease_time
my $pxe_lease_time = $optionFilter->pxe_lease_time();
# Modify pxe_lease_time
$optionFilter->pxe_lease_time("3600");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP option filter object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP Filter Option object

```
my  $optionfilter = Infoblox::DHCP::Filter::Option->new(

       name             => "option_filter",
       next_server      => "abc_server.com" ,
       boot_file        => "file1" ,
       boot_server      => "boot_server.com" ,
       lease_time       => "7200",
       pxe_lease_time   => "3600",
      );

unless ($optionfilter) {
   die("Construct DHCP Filter::Option failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Filter::Option object created successfully\n";
my $response = $session->add($optionfilter)
       or die("Add DHCP Filter Option failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "DHCP Filter Option added successfully\n";
```

#Search for a specific DHCP Filter Option

```
#Search the  DHCP Filter Option
my @retrieved_objs = $session->search(
    object             => "Infoblox::DHCP::Filter::Option",
    name               => "option.*",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search DHCP Filter Option  failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Search DHCP Filter Option found at least 1 matching entry\n";
```

#Get and modify a DHCP Filter Option attribute

```
#Get  DHCP Filter Option through the session
my  @retrieved_objs = $session->get(

              object      =>"Infoblox::DHCP::Filter::Option" ,
              name        => "option_filter",
                      );

my $object = $retrieved_objs[0];
unless ($object) {
    die("Get option filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Get DHCP Filter Option found at least 1 matching entry\n";

#Modify one of the attributes of the specified DHCP Filter Option attribute

$object->boot_server("another_Server");

#Apply the changes
$session->modify($object)
    or die("Modify DHCP Filter Option attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "DHCP Filter Option object modified successfully \n";
```

#Remove a DHCP Filter Option Object

```
 #Get object through the session
 my @retrieved_objs = $session->get(

           object  => "Infoblox::DHCP::Filter::Option" ,
           "name"  => "option_filter",
                   );

   my $object = $retrieved_objs[0];

   unless ($object) {
    die("Get a DHCP Filter Optionobject failed: ",
        $session->status_code() . ":" . $session->status_detail());
    }

print "Get a DHCP Filter Option object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove a DHCP Filter Option failed: ",
        $session->status_code() . ":" . $session->status_detail());

print "DHCP Filter Option removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Filter::RelayAgent - DHCP relay agent filter object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - comment( )
  - circuit_id_name( )
  - extensible_attributes( )
  - remote_id_name( )
  - is_circuit_id( )
  - is_remote_id( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Filter::RelayAgent - DHCP relay agent filter object

DESCRIPTION

The Infoblox appliance can screen address requests through relay agent filters (DHCP option 82) that assist the agents in forwarding address assignments across the proper circuit. When a relay agent receives the DHCPDISCOVER message, it can add one or two agent IDs in the DHCP option 82 suboption fields to the message. If the agent ID strings match those defined in a relay agent filter applied to a DHCP address range, the Infoblox appliance either assigns addresses from that range or denies the request (based on previously configured parameters; that is, the Grant lease and Deny lease parameters).

CONSTRUCTOR

```
  my $relayagentFilter = Infoblox::DHCP::Filter::RelayAgent->new(
                 name                => $string | $ipv4addr, #Required
                 comment             => $string,         #Optional / Default is empty
                 circuit_id_name      => $string,            #Optional / Default is empty
                 extensible_attributes => { $string => $string | $num, $string => [ $string | $num, ... ],
... }, #Optional / Default is undefined
                 remote_id_name       => $string,            #Optional / Default is empty
                 is_circuit_id        => $string,            #Optional / Default is empty
                 is_remote_id         => $string,            #Optional / Default is empty
                );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that can be applied to a DHCP relay agent filter object.

Infoblox::Session->add( )

Use this method to add a DHCP relay agent filter object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
       #Construct an object
       my  $relayagentFilter = Infoblox::DHCP::Filter::RelayAgent->new(
              name             => "relayagent_filter",
              circuit_id_name  => "circuit",
              remote_id_name   => "remote"
               );

       # Submit for adding a relay agent filter
       my $response = $session->add( $relayagentFilter );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching DHCP relay agent filter objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP relay agent filter object:

 name - Required. A relay agent filter name in string format or in ipv4address format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my  @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::Filter::RelayAgent" ,
    name   => "relayagent_filter",
);

# get all objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object                => "Infoblox::DHCP::Filter::RelayAgent",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP relay agent filter object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify circuit id name method.
$object->circuit_id_name("diff_Circuit");
# Submit modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP relay agent filter object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, then submit this object for removal.

Example

```
# Get the object under the same name
my @retrieved_objs = $session->get(
           object  => "Infoblox::DHCP::Filter::RelayAgent" ,
           "name"  => "relayagent_filter",
       );

# find the desired object from retrieved list.
my $desired_obj = $retrieved_objs[0];

# Submit for removal
my $response = $session->remove( $desired_obj);
```

Infoblox::Session->search( )

Use this method to search for a DHCP relay agent filter object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP Network object:

 name - Required. A relay agent filter name in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 comment - Optional . A comment in string format (regular expression).
```

Example

```
# search for all DHCP relayagent objects
     my @retrieved_objs = $session->search(
               object   => "Infoblox::DHCP::Filter::RelayAgent",
               name     => 'relayagent\..*',
```

```
                );

# search all DHCP relay agent associations with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DHCP::Filter::RelayAgent",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP relay agent filter object.

name( )

Use this method to set or retrieve the name of a DHCP relay agent filter object. The name can be an IP address or name of the router acting as relay agent.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the name in string format or in ipv4addr format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2). The default value for this field is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $name = $relayagentFilter->name()
# Modify name
$relayagentFilter->name("relay_filter");
```

comment( )

Use this method to set or retrieve the descriptive comment of a DHCP relay agent filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $relayagentFilter->comment();
#Modify comment
$relayagentFilter->comment("Modified DHCP relay agent filter object comment");
```

circuit_id_name( )

Use this method to set or retrieve a circuit_id_name of a DHCP relay agent filter object. This filter identifies the circuit between the remote host and the relay agent. For example, the identifier can be the ingress interface number of the circuit access unit, perhaps concatenated with the unit ID number and slot number. Also, the circuit ID can be an ATM virtual circuit ID or cable data virtual circuit ID.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify circuit id name in string format. example: Text String: 001122334455 Binary String: \x00\x11\x22\x33\x44\x55. The default value for this field is empty.

To reset this value, set it to an empty string together with the is_circuit_id flag (anything except "Matches_Value").

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get circuit_id_name
my $circuit_id_name = $relayagentFilter->circuit_id_name();
# Modify circuit_id_name
$relayagentFilter->circuit_id_name("001122334455");
# Set is_circuit_id to "Not_Set"
$relayagentFilter->is_circuit_id("Not_Set");
# Set circuit_id_name to empty string
$relayagentFilter->circuit_id_name("");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP relay agent filter object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $relayagentFilter->extensible_attributes();
#Modify extensible attributes
$relayagentFilter->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter',
'Tom' ] });
```

remote_id_name( )

Use this method to set or retrieve a remote ID name attribute of a relay agent filter object. This filter identifies the remote host. The remote ID name can represent many different things such as the caller ID telephone number for a dial-up connection, a user name for logging in to the ISP, a modem ID, etc. When the remote ID name is defined on the relay agent, the DHCP server will have a trusted relationship to identify the remote host. The remote ID name is considered as a trusted identifier.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify remote_id_name in string format. example: Text String: 001122334455 Binary String: \x00\x11\x22\x33\x44\x55. The default value for this field is empty.

To reset this value, set it to an empty string together with the is_remote_id flag (anything except "Matches_Value").

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get remote_id_name
my $remote_id_name = $relayagentFilter->remote_id_name();
# Modify remote_id_name
$relayagentFilter->remote_id_name("\x00\x11\x22\x33\x44\x55");
# Set is_remote_id to "Not_Set"
$relayagentFilter->is_remote_id("Not_Set");
# Set remote_id_name to empty string
$relayagentFilter->remote_id_name("");
```

is_circuit_id( )

Use this method to set or retrieve the is_circuit_id flag of a DHCP relay agent filter object. The possible values for the is_circuit_id flag are Any, Not_Set, Matches_Value. The circuit_id value takes effect only if the value is Matches_Value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify is_circuit_id in string format. example: Text String: Any

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get is_circuit_id
my $is_circuit_id = $relayagentFilter->is_circuit_id();
# Modify is_circuit_id
$relayagentFilter->is_circuit_id("Any");
```

is_remote_id( )

Use this method to set or retrieve the is_remote_id flag of a DHCP relay agent filter object. The possible values for the is_remote_id flag are Any, Not_Set, Matches_Value. The remote_id value takes effect only if the value is Matches_Value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify is_remote_id in string format. example: Text String: Not_Set

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get is_remote_id
my $is_remote_id = $relayagentFilter->is_remote_id();
# Modify is_remote_id
$relayagentFilter->is_remote_id("Not_Set");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP relay agent filter object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a FilterRule relayagent filter object>

my $filter = Infoblox::DHCP::FilterRule::RelayAgent->new(
               filter_name => "relayagent_filter",
               permission => "grant" );
```

#Create a DHCP relay agent filter object

```
my  $relayagentfilter = Infoblox::DHCP::Filter::RelayAgent->new(

       name             => "relayagent_filter",
       circuit_id_name  => "circuit",
       remote_id_name   => "remote"
      );
my $response = $session->add($relayagentfilter)
       or die("Add DHCP Relay Agent Filter failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "DHCP Relay Agent Filter added successfully\n";
```

#Search for a specific DHCP relay agent filter

```
#Search the DHCP Relay Agent Filter
my @retrieved_objs = $session->search(
               object => "Infoblox::DHCP::Filter::RelayAgent",
               name   => "relayagent.*",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search DHCP Relay Agent Filter  failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Search DHCP relay agent filter found at least 1 matching entry\n";
```

#Get and modify a DHCP relay agent filter attribute

```
#Get  DHCP Relay Agent Filter through the session
my  @retrieved_objs = $session->get(
              object => "Infoblox::DHCP::Filter::RelayAgent" ,
              name   => "relayagent_filter",
        );
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Get DHCP relay agent filter found at least 1 matching entry\n";

#Modify the attributes of the specified DHCP Relay Agent Filter object
$object->circuit_id_name("diff_Circuit");
$object->remote_id_name("diff_remote");
$session->modify($object)
    or die("Modify DHCP Relay Agent Filter attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "DHCP Relay Agent Filter object modified successfully \n";

#Create the Network object with the member
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    comment => "add network",
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
    or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "Network object added to Infoblox appliance successfully\n";
```

#Create a DHCP Range object that uses the relayagent filter

```
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr                => "10.0.0.10",
        network                 => "10.0.0.0/8",
        start_addr              => "10.0.0.1",
        authority               => "true",
        filters                 => [$filter],
```

```
);
unless($dhcp_range) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}

print "DHCP Range object created successfully\n";

#Add the DHCP Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to the network successfully\n";

#Remove the Network object into the Infoblox appliance through a session
$session->remove($network)
    or die("Remove Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network object removed from Infoblox appliance successfully\n";
```

#Remove a DHCP relay agent filter Object

```
 #Get object through the session
 my @retrieved_objs = $session->get(
         object   => "Infoblox::DHCP::Filter::RelayAgent" ,
         name     => "relayagent_filter",
       );
  my $object = $retrieved_objs[0];    unless ($object) {
    die("Get a DHCP Relay Agent Filterobject failed: ",
        $session->status_code() . ":" . $session->status_detail());
  }

print "Get a DHCP relay agent filter object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove a DHCP Relay Agent Filter failed: ",
        $session->status_code() . ":" . $session->status_detail());

print "DHCP relay agent filter removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::Network, Infoblox::DHCP::Range, Infoblox::DHCP::FilterRule::RelayAgent, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::FilterRule::MAC - DHCP MAC Address Filter Rule object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::Range->filters( )
- METHODS
  - filter_name( )
  - permission( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::FilterRule::MAC - DHCP MAC Address Filter Rule object.

DESCRIPTION

MAC Address Filter Rule allows a user to grant/deny the lease of an IP defined in DHCP Range.

CONSTRUCTOR

```
my $mac_filter_rule = Infoblox::DHCP::FilterRule::MAC->new(
      filter_name   => $string,          #Required
      permission    => "grant" | "deny", #Required
);
```

MODULE METHODS

The following functions can be applied to a DHCP MAC Filter Rule object.

Infoblox::DHCP::Range->filters( )

Use this method to add a DHCP MAC Address Filter Rule object to the DHCP Range on an Infoblox appliance. See Infoblox::DHCP::Range->filters() for parameters and return values.

Example

```
#Construct a DHCP MAC Address Filter Rule object
my $mac_filter_rule1 = Infoblox::DHCP::FilterRule::MAC->new(
      filter_name   => "filter1",
      permission    => "grant",
);

#Add MAC Address Filter Rule to the DHCP Range object.
my $response = $dhcp_range->filters([$mac_filter_rule1]);
```

METHODS

This section describes all the methods that can be used to set and retrieve the attribute values of a DHCP MAC Address Filter Rule object.

filter_name( )

Use this method to set or retrieve the filter_name of a DHCP MAC Address Filter Rule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the DHCP MAC Address filter. Maximum length up to 1024 bytes is supported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get filter_name
my $filter_name = $mac_filter_rule->filter_name();
#Modify filter_name
$mac_filter_rule->filter_name("filter2");
```

permission( )

Use this method to set or retrieve the permission of a DHCP MAC Address Filter Rule object.

Permission indicates whether members of this MAC address filter are allowed or denied an IP address from the defined DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Permission value can be either "grant" or "deny".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get permission
my $permission = $mac_filter_rule->permission();
#Modify permission
$mac_filter_rule->permission("deny");
```

SAMPLE CODE

The following sample code demonstrates different functions that can be applied to an object such as modify. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP MAC Address Filter Rule object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the Network object
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    comment => "add network",
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
    or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network object added to Infoblox appliance successfully\n";

#Construct a DHCP MAC Address Filter object
```

```
my $mac_filter = Infoblox::DHCP::Filter::MAC->new(
       name => "filter1",
);

unless ( $mac_filter ) {
       die("Construct MAC Address Filter failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail() );
}
print "MAC Address Filter object created successfully\n";

#Add the MAC Address Filter to the Infoblox appliance through a session
$session->add($mac_filter)
    or die("Add MAC Address Filter object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "MAC Address Filter object added to Infoblox appliance successfully\n";
```

#Create a DHCP MAC Address Filter Rule object

```
#Construct a DHCP MAC Address Filter Rule object
my $mac_filter_rule1 = Infoblox::DHCP::FilterRule::MAC->new(
      filter_name   => "filter1",
      permission    => "grant",
);

unless ( $mac_filter_rule1 ) {
       die("Construct MAC Address Filter Rule failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail() );
}
print "MAC Address Filter Rule object created successfully\n";

#Create the DHCP Range object with this MAC Address Filter Rule.
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr   => "10.0.0.10",
        network    => "10.0.0.0/8",
        start_addr => "10.0.0.1",
        filters    => [ $mac_filter_rule1 ],
);
unless($dhcp_range) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

#Add the DHCP Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to Infoblox appliance successfully\n";

B<#Get and modify a MAC Address Filter Rule object>

#Get MAC Address Filter Rule object from the DHCP Range object through session
my @retrieved_objs = $session->get(
    object      => "Infoblox::DHCP::Range",
    start_addr  => "10.0.0.1",
);

my $object = $retrieved_objs[0];
my $filter = $object->filters();

my @filters = @{$filter};

my $mac_filterrule = $filters[0];

#Modify one of the attributes of the obtained MAC Address Filter Rule object
$mac_filterrule->permission("deny");
$object->filters([$mac_filterrule]);

#Apply the changes
$session->modify($object)
    or die("Modify MAC Address Filter Rule object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "MAC Address Filter Rule object modified and added to the DHCP Range object successfully \n";
```

#Remove a MAC Address Filter Rule object

```
$object->filters([]);
```

```
my $response = $session->modify($object);
unless($response) {
       die("Remove MAC Address Filter Rule failed: ",
               session->status_code() . ":" . session->status_detail() );
}
print "MAC Address Filter Rule object removed successfully \n";

#Submit the network object for removal
$session->remove($network)
    or die("Remove Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

#Submit the mac address filter object for removal
$session->remove($mac_filter)
    or die("Remove MAC Address Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "MAC Address Filter object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::MAC, Infoblox::DHCP::FilterRule::RelayAgent, Infoblox::DHCP::Network, Infoblox::DHCP::Range, Infoblox::Session->add(),Infoblox::Session->modify(), Infoblox::Session->remove(),Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::FilterRule::NAC - DHCP NAC Filter Rule object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::Range->filters( )
- METHODS
  - filter_name( )
  - permission( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::FilterRule::NAC - DHCP NAC Filter Rule object.

DESCRIPTION

A NAC Filter Rule object is used to grant or deny the lease of an IP address in a DHCP range based on whether the authentication response matches the NAC filter.

CONSTRUCTOR

```
my $nac_filter_rule = Infoblox::DHCP::FilterRule::NAC->new(
      filter_name   => $string,          #Required
      permission    => "grant" | "deny", #Required
);
```

MODULE METHODS

The following functions can be applied to a DHCP NAC Filter Rule object.

Infoblox::DHCP::Range->filters( )

Use this method to add a DHCP NAC Filter Rule object to a DHCP range. See Infoblox::DHCP::Range->filters() for parameters and return values.

Example

```
#Construct a DHCP NAC Filter Rule object
my $nac_filter_rule1 = Infoblox::DHCP::FilterRule::NAC->new(
      filter_name   => "filter1",
      permission    => "grant",
);

#Add a NAC Filter Rule to the DHCP Range object.
my $response = $dhcp_range->filters([$nac_filter_rule1]);
```

METHODS

This section describes all the methods that can be used to set and retrieve the attribute values of a DHCP NAC Filter Rule object.

filter_name( )

Use this method to set or retrieve the filter_name of a DHCP NAC Filter Rule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the DHCP NAC filter. The maximum length supported is 1024 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get filter_name
my $filter_name = $nac_filter_rule->filter_name();
#Modify filter_name
$nac_filter_rule->filter_name("filter2");
```

permission( )

Use this method to set or retrieve the permission of a DHCP NAC Filter Rule object.

The permission indicates whether a DHCP client whose NAC authentication response matches this NAC filter is allowed or denied a lease from the defined DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The permission value can be either "grant" or "deny".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get permission
my $permission = $nac_filter_rule->permission();
#Modify permission
$nac_filter_rule->permission("deny");
```

SAMPLE CODE

The following sample code demonstrates different functions that can be applied to an object such as modify. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP NAC Filter Rule object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the Network object
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    comment => "add network",
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
    or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network object added to Infoblox appliance successfully\n";
```

```
#Construct a DHCP NAC Filter object
my $nac_filter = Infoblox::DHCP::Filter::NAC->new(
       name => "filter1",
);

unless ( $nac_filter ) {
       die("Construct NAC Filter failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail() );
}
print "NAC Filter object created successfully\n";

#Add the NAC Filter to the Infoblox appliance through a session
$session->add($nac_filter)
    or die("Add NAC Filter object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "NAC Filter object added to Infoblox appliance successfully\n";
```

#Create a DHCP NAC Filter Rule object

```
#Construct a DHCP NAC Filter Rule object
my $nac_filter_rule1 = Infoblox::DHCP::FilterRule::NAC->new(
      filter_name   => "filter1",
      permission    => "grant",
);

unless ( $nac_filter_rule1 ) {
       die("Construct NAC Filter Rule failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail() );
}
print "NAC Filter Rule object created successfully\n";

#Create the DHCP Range object with this NAC Filter Rule.
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr   => "10.0.0.10",
        network    => "10.0.0.0/8",
        start_addr => "10.0.0.1",
        filters    => [ $nac_filter_rule1 ],
);
unless($dhcp_range) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

#Add the DHCP Range object through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to Infoblox appliance successfully\n";

B<#Get and modify a NAC Filter Rule object>

#Get NAC Filter Rule object from the DHCP Range object through session
my @retrieved_objs = $session->get(
    object      => "Infoblox::DHCP::Range",
    start_addr  => "10.0.0.1",
);

my $object = $retrieved_objs[0];
my $filter = $object->filters();

my @filters = @{$filter};

my $nac_filterrule = $filters[0];

#Modify one of the attributes of the obtained NAC Filter Rule object
$nac_filterrule->permission("deny");
$object->filters([$nac_filterrule]);

#Apply the changes
$session->modify($object)
    or die("Modify NAC Filter Rule object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "NAC Filter Rule object modified and added to the DHCP Range object successfully \n";
```

#Remove a NAC Filter Rule object

```
$object->filters([]);
```

```
my $response = $session->modify($object);
unless($response) {
       die("Remove NAC Filter Rule failed: ",
               session->status_code() . ":" . session->status_detail() );
}
print "NAC Filter Rule object removed successfully \n";

#Submit the network object for removal
$session->remove($network)
    or die("Remove Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

#Submit the NAC Address Filter object for removal
$session->remove($nac_filter)
    or die("Remove NAC Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "NAC Filter object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::FilterRule::RelayAgent, Infoblox::DHCP::Network, Infoblox::DHCP::Range, Infoblox::Session->add(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::FilterRule::Option - DHCP filter rule option object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::Range->filters( )
- METHODS
  - filter_name( )
  - permission( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::FilterRule::Option - DHCP filter rule option object.

DESCRIPTION

A DHCP filter rule option object gives the ability to grant and/or deny a lease of an IP defined in DHCP Range.

CONSTRUCTOR

```
my $FilterRuleOption = Infoblox::DHCP::FilterRule::Option->new(
             filter_name  =>  $string,          #Required
             permission   =>  "grant" | "deny", #Required
         );
```

MODULE METHODS

The following functions are available to apply to a DHCP filter rule option object.

Infoblox::DHCP::Range->filters( )

Use this function to specify filters at the DHCP range level. See details for Infoblox::DHCP::Range objects

Example

```
#Construct DHCP filter rule option object
my $filter = Infoblox::DHCP::FilterRule::Option->new(
    filter_name => "filter1",
    permission  => "grant"
);

# Configure  filters
my $response = $range->filters([$res]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP filter rule option object.

filter_name( )

Use this method to set or retrieve the filter name of a DHCP filter rule option object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify filter name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get filters from  range
my $resp = $range->filters();
# Modify filters
$range->filters([$res]);
```

permission( )

Use this method to set or retrieve the permission of a DHCP filter rule option object. When selecting Grant lease, you need to assign addresses from the address range to requesting hosts whose user class (as defined on their network adapter) matches an entry in the option filter rules. When selecting Deny lease, you need to refuse an address request from a host whose user class matches an entry in the option filter rules.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The permission can be either "grant" or "deny". The default value is "grant" lease.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get permission
 $res->permission();
#modify permission value
$res->permission("deny");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, modify. Also, this sample includes error handling for the operations.

##Preparation prior to a DHCP filter rule option object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

  #Create a DHCP Member object
   my $memberdhcp = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2"
   );
   unless($memberdhcp) {
       die("Construct member failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
   }
   print "DHCP Member object created successfully\n";

#Create the Network object with the member
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    comment => "add network",
    members => [ $memberdhcp ]
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
}
print "Network object created successfully\n";
```

#Add the Network object into the Infoblox appliance through a session

```
$session->add($network)
    or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network object added to Infoblox appliance successfully\n";

my $opt1 = Infoblox::DHCP::Option->new(
      name => "user-class" ,
      value => "77"
   );
```

#Create the Option Filter object

```
my $option_filter = Infoblox::DHCP::Filter::Option->new(
      name => 'filter1' ,
      option_list => [$opt1],

);

unless($option_filter) {
       die("Construct Option Filter object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Option Filter object created successfully\n";
```

#Add the filter rule option object into the Infoblox appliance

```
$session->add($option_filter)
    or die("Add option Filter object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Option Filter object added to Infoblox appliance successfully\n";
```

#Construct DHCP Filter rule Option object

```
my $filter = Infoblox::DHCP::FilterRule::Option->new(
    filter_name => "filter1",
    permission  => "grant"
);
```

#Create a DHCP Range object

```
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr     => "10.0.0.10",
        network      => "10.0.0.0/8",
        start_addr   => "10.0.0.1",
        authority    => "true",
        comment      => "add range",
        filters      => [$filter],
        member       => $memberdhcp,
);
unless($dhcp_range) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

#Add the DHCP Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to the network successfully\n";
```

#Get and modify the filter rule option object

```
#get  object from session
my @result = $session->get(
    object => "Infoblox::DHCP::Range",
    start_addr => "10.0.0.1"
);
unless (scalar(@result) == 0) {
    my $range = $result[0];

    if ($range) {
```

```
    my $resp = $range->filters();
    my @arr = @{$resp};
    my $res = $arr[0];

        #modify permission value
        $res->permission("deny");
        $range->filters([$res]);
        #update session
        $session->modify($range)
            or die("modify failed: ",
               $session->status_code(), $session->status_detail());

            print "Modify filter permisson successful\n";
                      }
                    } else {
             print "No range found.";
                    }
             $session->remove($option_filter);
             print "Remove filter1 successfull\n";

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::Range, Infoblox::DHCP::Network, Infoblox::DHCP::Member, Infoblox::DHCP::Option, Infoblox::Session->get(), Infoblox::Session->add(),

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::FilterRule::RelayAgent - DHCP RelayAgent Filter Rule object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::Range->filters( )
- METHODS
  - filter_name( )
  - permission( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::FilterRule::RelayAgent - DHCP RelayAgent Filter Rule object.

DESCRIPTION

A relay agent adds the circuit ID and Remote ID information to the client request and sends it to DHCP Server. Infoblox appliance uses Relay Agent Filter Rules to grant/deny the lease to a request that comes from those Relay Agents.

CONSTRUCTOR

```
my $ra_filter_rule = Infoblox::DHCP::FilterRule::RelayAgent->new(
      filter_name   => $string,          #Required
      permission    => "grant" | "deny", #Required
);
```

MODULE METHODS

The following functions are available to apply to a DHCP RelayAgent Filter Rule object.

Infoblox::DHCP::Range->filters( )

Use this method to add a DHCP RelayAgent Filter Rule object to the DHCP Range on the Infoblox appliance. See Infoblox::DHCP::Range->filters() for parameters and return values.

Example

```
#Construct a DHCP RelayAgent Filter Rule object
my $ra_filter_rule1 = Infoblox::DHCP::FilterRule::RelayAgent->new(
      filter_name   => "filter1",
      permission    => "grant",
);

#Add RelayAgent Filter Rule to the DHCP Range object.
my $response = $dhcp_range->filters([$ra_filter_rule1]);
```

METHODS

This section describes all the methods that can be used to set and retrieve the attribute values of a DHCP RelayAgent Filter Rule object.

filter_name( )

Use this method to set or retrieve the filter_name of a DHCP RelayAgent Filter Rule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the DHCP RelayAgent filter. Maximum length up to 1024 bytes is supported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get filter_name
my $filter_name = $ra_filter_rule->filter_name();
#Modify filter_name
$ra_filter_rule->filter_name("filter2");
```

permission( )

Use this method to set or retrieve the permission of a DHCP RelayAgent Filter Rule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the permission in string format. The permission can be either "grant" or "deny".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get permission
my $permission = $ra_filter_rule->permission();
#Modify permission
$ra_filter_rule->permission("deny");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as modify. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP RelayAgent Filter Rule object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the Network object
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    comment => "add network",
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
    or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network object added to Infoblox appliance successfully\n";

#Construct a DHCP RelayAgent Filter object
my $ra_filter = Infoblox::DHCP::Filter::RelayAgent->new(
       name            => "filter1",
```

```
        circuit_id_name => "circuit",
        remote_id_name  => "remote"
);

unless ( $ra_filter ) {
        die("Construct RelayAgent Filter failed: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail() );
}
print "RelayAgent Filter object created successfully\n";

#Add the RelayAgent Filter to the Infoblox appliance through a session
$session->add($ra_filter)
    or die("Add RelayAgent Filter object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "RelayAgent Filter object added to Infoblox appliance successfully\n";
```

#Create a DHCP RelayAgent Filter Rule object

```
#Construct a DHCP RelayAgent Filter Rule object
my $ra_filter_rule1 = Infoblox::DHCP::FilterRule::RelayAgent->new(
      filter_name   => "filter1",
      permission    => "grant",
);

unless ( $ra_filter_rule1 ) {
        die("Construct RelayAgent Filter Rule failed: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail() );
}
print "RelayAgent Filter Rule object created successfully\n";

#Create the DHCP Range object with this RelayAgent Filter Rule.
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr   => "10.0.0.10",
        network    => "10.0.0.0/8",
        start_addr => "10.0.0.1",
        filters    => [ $ra_filter_rule1 ],
);
unless($dhcp_range) {
        die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

#Add the DHCP Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to Infoblox appliance successfully\n";

B<#Get and modify a RelayAgent Filter Rule object>

#Get RelayAgent Filter Rule object from the DHCP Range object through session
my @retrieved_objs = $session->get(
    object      => "Infoblox::DHCP::Range",
    start_addr  => "10.0.0.1",
);

my $object = $retrieved_objs[0];
my $filter = $object->filters();

my @filters = @{$filter};

my $ra_filterrule = $filters[0];

#Modify one of the attributes of the obtained RelayAgent Filter Rule object
$ra_filterrule->permission("deny");
$object->filters([$ra_filterrule]);

#Apply the changes
$session->modify($object)
    or die("Modify RelayAgent Filter Rule object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "RelayAgent Filter Rule object modified and added to the DHCP Range object successfully \n";
```

#Remove a RelayAgent Filter Rule object

```
$object->filters([]);
```

```
my $response = $session->modify($object);
unless($response) {
       die("Remove RelayAgent Filter Rule failed: ",
               session->status_code() . ":" . session->status_detail() );
}
print "RelayAgent Filter Rule object removed successfully \n";

#Submit the network object for removal
$session->remove($network)
    or die("Remove Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

#Submit the relay agent filter object for removal
$session->remove($ra_filter)
    or die("Remove RelayAgent Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "RelayAgent Filter object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::MAC, Infoblox::DHCP::FilterRule::RelayAgent, Infoblox::DHCP::Network, Infoblox::DHCP::Range, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(),Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::FixedAddr - DHCP Fixed Address object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - bootfile( )
  - agent_circuit_id( )
  - agent_remote_id( )
  - always_update_dns( )
  - bootserver( )
  - client_identifier_prepend_zero( )
  - comment( )
  - configure_for_dhcp( )
  - ddns_domainname( )
  - ddns_hostname( )
  - deny_bootp( )
  - dhcp_client_identifier( )
  - disable( )
  - discovered_name( )
  - discoverer( )
  - enable_ddns( )
  - extensible_attributes( )
  - first_discovered( )
  - ignore_dhcp_option_list_request( )
  - ipv4addr( )
  - last_discovered( )
  - mac( )
  - match_client( )
  - member( )
  - name( )
  - netbios( )
  - ms_options( )
  - network( )
  - network_component_description( )
  - network_component_ip( )
  - network_component_name( )
  - network_component_port_description( )
  - network_component_port_name( )
  - network_component_port_number( )
  - network_component_type( )
  - network_view( )
  - nextserver( )
  - options( )
  - os( )
  - override_ddns_domainname( )
  - port_duplex( )
  - port_link_status( )
  - port_speed( )
  - port_status( )
  - port_vlan_description( )
  - port_vlan_name( )
  - port_vlan_number( )
  - pxe_lease_time( )
  - template( )
  - v_cluster( )
  - v_datacenter( )
  - v_host( )
  - v_name( )
  - v_netadapter( )
  - v_switch( )
  - v_type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO

COPYRIGHT

NAME

Infoblox::DHCP::FixedAddr - DHCP Fixed Address object.

DESCRIPTION

A Fixed Address is a specific IP address that a DHCP server always assigns when a lease request comes from a particular MAC address of the client.

CONSTRUCTOR

```
 my $fixed_addr = Infoblox::DHCP::FixedAddr -> new(
       ipv4addr                          => $ipv4addr,                    #Required
       agent_circuit_id                  => $circuit_id,                  #Optional / Default is undefined
       agent_remote_id                   => $remote_id,                   #Optional / Default is undefined
       always_update_dns                 => "true" | "false",             #Optional / Default is "false"
       bootfile                          => $filename | undef,            #Optional / Default is undefined
       bootserver                        => $ipv4addr | $fqdn | undef,    #Optional / Default is undefined
       client_identifier_prepend_zero    => "true" | "false",             #Optional / Default is "false"
       comment                           => $string,                      #Optional / Default is empty
       configure_for_dhcp                => "true" | "false",             #Optional / Default is "false"
                                                                          # When both the IP address and MAC
address are present, the default value is "true"
                                                                          # Applicable only when adding this
fixed address to a host
       ddns_domainname                   => $string,                      #Optional / Default is empty
       ddns_hostname                     => $string,                      #Optional / Default is empty
       deny_bootp                        => "true" | "false" | undef,     #Optional / Default is undefined
       dhcp_client_identifier            => $string,                      #Optional / Default is empty
       disable                           => "true" | "false",             #Optional / Default is "false"
       enable_ddns                       => "true" | "false",             #Optional / Default is "false"
       extensible_attributes             => { $string => $string | $num, $string => [ $string | $num, ... ],
... } #Optional / Default is undefined
       ignore_dhcp_option_list_request => "true" | "false" | undef,     #Optional / Default is "false"
       last_discovered                   => $string,                      #Optional / Default is empty
       mac                               => $mac_addr,                    #Optional / Default is empty
       match_client                      => $string,                      #Optional / Default is "MAC"
                                                                          #Possible values are:
                                                                          #
"MAC","CLIENT_IDENTIFIER","RESERVED",
                                                                          # "CIRCUIT_ID", "REMOTE_ID"
       member                            => $msmember,                    #Optional
       ms_options                        => [$msopt1, $msopt2, ...]       #Optional / Default is empty
       name                              => $name,                        #Optional
       network                           => $network,                     #Optional
       network_view                      => $NetworkView,                 #Optional / Default is "default"
network view
       nextserver                        => $ipv4addr | $fqdn | undef,    #Optional / Default is undefined
       options                           => [$option1, $option2, ...]     #Optional / Default is empty
       override_ddns_domainname          => "true" | "false",             #Optional / Default value is "false"
       pxe_lease_time                    => $time,                        #Optional / Default is empty
       template                          => $string,                      #Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DHCP Fixed Address object.

Infoblox::Session->add( )

Use this method to add a DHCP Fixed Address object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
       #Construct a DHCP Fixed Address object
       my $fixed_addr = Infoblox::DHCP::FixedAddr->new(
              "ipv4addr"                          => "10.0.0.5",
              "bootfile"                          => "bootfile",
              "bootserver"                        => "bootserver.com",
              "client_identifier_prepend_zero"    => "true",
```

```
        "comment"                          => "add fixed address using uid",
        "ddns_domainname"                  => "ddns.domain.com",
        "ddns_hostname"                    => "ddns.host.domain.com",
        "deny_bootp"                       => "false",
        "dhcp_client_identifier"           => "foo",
        "disable"                          => "false",
        "enable_ddns"                      => "false",
        "match_client"                     => "MAC",
        "network"                          => "10.0.0.0/8",
        "network_view"                     => $nview,
        "nextserver"                       => "3.3.3.3",
        "options"                          => [ $option1,  $option2, $option3,  $option4, $option5,
$option6 ],
        "pxe_lease_time"                   => "3600",
 );

 #Submit for addition
 my $response = $session->add($fixed_addr)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
 Apply the following attributes to get a specific DHCP Fixed Address object:

  ipv4addr                - Optional. IPV4 address of the fixed address.
  mac                     - Optional. MAC address of the fixed address.
  dhcp_client_identifier  - Optional, DHCP option 61 client identifier.
  network                 - Optional. Network address of the fixed address.
  network_view            - Optional. Network view in which the fixed address is located. By default,
all network views are searched.
  extensible_attributes   - Optional. A hash reference containing extensible attributes.
  v_cluster               - Optional. Discovered VMware cluster name.
  v_datacenter            - Optional. Discovered VMware datacenter name.
  v_host                  - Optional. Discovered VMware server name.
  v_name                  - Optional. Discovered VMware host or virtual machine name.
  v_netadapter            - Optional. Discovered VMware physical network adapter name.
  v_switch                - Optional. Discovered VMware virtual switch name.
  v_type                  - Optional. Discovered VMware entity type (possible values:
"VirtualMachine", "HostSystem", "VirtualCenter").
```

Examples

```
#Get DHCP Fixed Address object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FixedAddr",
    mac    => "11:11:11:11:11.11",
);
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FixedAddr",
    dhcp_client_identifier => "foo",
);
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::FixedAddr",
    ipv4addr     => "10.0.0.4",
    network_view => "default",
);

my @retrieved_objs = $session->get(
   object                => "Infoblox::DHCP::FixedAddr",
   extensible_attributes => { 'Site' => 'Santa Clara' });

 # get FixedAddr by v_type:
 my @retrieved_objs = $session->get(
   object    => 'Infoblox::DHCP::FixedAddr',
   v_type    => 'VirtualMachine');
```

Infoblox::Session->modify( )

Use this method to modify a DHCP Fixed Address object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the pxe_lease_time
$fixed_address->pxe_lease_time("1234");
#Submit modification
```

```
my $response = $session->modify( $fixed_address );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP Fixed Address object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP Fixed Address object, and then submit this object for removal.

Example

```
#Get DHCP Fixed Address object through the session
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::FixedAddr",
    mac          => "11:11:11:11:11:11",
    network_view => "default"
);
#Find the desired object from the retrieved list.
my $desired_fixed_addr = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_fixed_addr );
```

Infoblox::Session->search( )

Use this method to search for DHCP Fixed Address object and matching Host object with MAC address in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP Fixed Address object:

  ipv4addr      - Optional. IPV4 address of the fixed address (regular expression).
  mac           - Optional. MAC address of the fixed address (regular expression).
  network       - Optional. Network address of the fixed address.
  network_view  - Optional. Network view in which the fixed address is located. By default, all
network views are searched.
  extensible_attributes - Optional. A hash reference containing extensible attributes.
  v_cluster     - Optional. Discovered VMware cluster name (regular expression).
  v_datacenter  - Optional. Discovered VMware datacenter name (regular expression).
  v_host        - Optional. Discovered VMware server name (regular expression).
  v_name        - Optional. Discovered VMware host or virtual machine name (regular expression).
  v_netadapter  - Optional. Discovered VMware physical network adapter name (regular expression).
  v_switch      - Optional. Discovered VMware virtual switch name (regular expression).
  v_type        - Optional. Discovered VMware entity type (possible values: "VirtualMachine",
"HostSystem", "VirtualCenter").
  no_hostaddresses - Optional. If specified and set to "true", only Fixed Address objects will be
returned by the search.
```

Examples

```
# Search for all DHCP Fixed Address objects that are specified in the network "10.0.0.0/8" in the
"default" network view.
my @retrieved_objs = $session->search(
    object       => "Infoblox::DHCP::FixedAddr",
    mac          => "11:11:11:11:11.*",
    network      => "10.0.0.0/8",
    network_view => "default"
);

# Search for all DHCP Fixed Address objects with "Santa Clara" for the "Site" extensible attribute
my @retrieved_objs = $session->search(
    object                => "Infoblox::DHCP::FixedAddr",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);

# search FixedAddr by v_datacenter:
my @retrieved_objs = $session->search(
   object                => 'Infoblox::DHCP::FixedAddr',
   v_datacenter => 'test.*');
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a DHCP Fixed Address object.

bootfile( )

Use this method to set or retrieve the bootfile value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::FixedAddr->bootserver() and Infoblox::DHCP::FixedAddr->nextserver().

Parameter

The name of the file that the client must download. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootfile
my $bootfile = $fixed_addr->bootfile();
#Modify bootfile
$fixed_addr->bootfile("boot_file");
#Un-override bootfile
$fixed_addr->bootfile(undef);
```

agent_circuit_id( )

Use this method to set or retrieve the agent circuit ID of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Use the agent_circuit_id to assign fixed addresses to DHCP clients from whom the member DHCP server receives DHCPDISCOVER and DHCPREQUEST messages that include option 82. To enable the member DHCP server to match the agent circuit ID, set the "match_client" field to "CIRCUIT_ID".

Returns

If you specified a parameter, the method returns true when the modification succeeds and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get agent_circuit_id
my $uid = $fixed_addr->agent_circuit_id();
#Modify agent_circuit_id
$fixed_addr->agent_circuit_id("foo");
```

agent_remote_id( )

Use this method to set or retrieve the agent remote ID of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Use the agent_remote_id to assign fixed addresses to DHCP clients from whom the member DHCP server receives DHCPDISCOVER and DHCPREQUEST messages that include option 82. To enable the member DHCP server to match the agent remote ID, set the "match_client" field to "REMOTE_ID".

Returns

If you specified a parameter, the method returns true when the modification succeeds and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get agent_remote_id
my $uid = $fixed_addr->agent_remote_id();
#Modify agent_remote_id
$fixed_addr->agent_remote_id("foo");
```

always_update_dns( )

Use this method to set or retrieve the always_update_dns flag of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This flag only applies if the fixed address is associated with a Microsoft (r) server. If it is set to "true", the DHCP server will always update DNS, regardless of the requests from the DHCP clients. If this flag is set to false, clients will be allowed to update DNS when they request it.

Parameter

Specify "true" to set the always_update_dns flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get always_update_dns
my $always_update_dns = $fixed_addr->always_update_dns();
#Modify always_update_dns
$fixed_addr->always_update_dns("false");
```

bootserver( )

Use this method to set or retrieve the bootserver of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::FixedAddr->bootfile() and Infoblox::DHCP::FixedAddr->nextserver().

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootserver
my $bootserver = $fixed_addr->bootserver();
#Modify bootserver
$fixed_addr->bootserver("bootserver2.com");
#Un-override bootserver
$fixed_addr->bootserver(undef);
```

client_identifier_prepend_zero( )

Use this method prepend "\000" to the dhcp-client-identifier of a fixed address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

When you set the client_identifier_prepend_zero parameter to "true", it prefixes a 0 (zero) to the DHCP client identifier whenever option 61 is used to assign fixed addresses. The following is an excerpt from "man dhcp-options":

Please be aware that some DHCP clients, when configured with client identifiers that are ASCII text, will prepend a zero to the ASCII text. So you may need to write:

```
option dhcp-client-identifier "\000foo"
```

rather than:

```
option dhcp-client-identifier "foo".
```

Returns

If you specified a parameter, the method returns true when the modification succeeds and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get client_identifier_prepend_zero
```

```
my $client_identifier_prepend_zero = $fixed_addr->client_identifier_prepend_zero();
#Modify dhcp_client_identifier
$fixed_addr->client_identifier_prepend_zero("true");
```

comment( )

Use this method to set or retrieve the descriptive comment of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $fixed_addr->comment();
#Modify comment
$fixed_addr->comment("Modified DHCP Fixed Address object comment");
```

configure_for_dhcp( )

Use this method to set or retrieve the configure_for_dhcp flag of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the configure_for_dhcp flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get configure_for_dhcp
my $configure_for_dhcp = $fixed_addr->configure_for_dhcp();
#Modify configure_for_dhcp
$fixed_addr->configure_for_dhcp("true");
```

ddns_domainname( )

Use this method to set or retrieve the DDNS domain name value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_dddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

Specify the DDNS domain name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the DDNS domain name
my $ddns_domainname = $fixed_addr->ddns_domainname();
#Modify the DDNS domain name
```

```
$fixed_addr->ddns_domainname("ddns.domain.com");
```

ddns_hostname( )

Use this method to set or retrieve the DDNS host name value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the DDNS host name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the DDNS host name
my $ddns_hostname = $fixed_addr->ddns_hostname();
#Modify the DDNS host name
$fixed_addr->ddns_hostname("host.ddns.domain.com");
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp flag of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_bootp flag or "false" to deactivate/unset it. The default value is undefined which indicates that this attribute inherits the upper-level setting

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $fixed_addr->deny_bootp();
#Modify deny_bootp
$fixed_addr->deny_bootp("true");
```

dhcp_client_identifier( )

Use this method to set or retrieve the dhcp-client-identifier value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Use the dhcp_client_identifier or uid to assign fixed addresses if the DHCP clients send option 61. A DHCP client identifier can be a string or a MAC address. Set the "match_client" field to "CLIENT_IDENTIFIER" for fixed addresses to use the dhcp_client_identifier instead of the mac_address.

Returns

If you specified a parameter, the method returns true when the modification succeeds and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dhcp_client_identifier
my $uid = $fixed_addr->dhcp_client_identifier();
#Modify dhcp_client_identifier
$fixed_addr->dhcp_client_identifier("foo");
or
$fixed_addr->dhcp_client_identifier("1:11::11::11::11::11::12");
```

disable( )

Use this method to set or retrieve the disable flag of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $fixed_addr->disable();
#Modify disable
$fixed_addr->disable("true");
```

discovered_name( )

Use this method to retrieve the discovered name of a DHCP Fixed Address object.

Parameter

none

Returns

The method returns the discovered name.

Example

```
#Get discovered_name
my $discovered_name = $fixed_addr->discovered_name();
```

discoverer( )

Use this method to retrieve the discoverer of a DHCP Fixed Address object.

Parameter

none

Returns

The method returns the discoverer value.

Example

```
#Get discoverer
my $discoverer = $fixed_addr->discoverer();
```

enable_ddns( )

Use this method to set or retrieve the enable_ddns flag of a DHCP Fixed Address object.

If enable_ddns value is specified as "true", then the A and PTR records corresponding to the Fixed Address object will be placed on the DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_ddns flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ddns
my $enable_ddns = $fixed_addr->enable_ddns();
#Modify enable_ddns
```

```
$fixed_addr->enable_ddns("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP Fixed Address object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $fixed_addr->extensible_attributes();
#Modify extensible attributes
$fixed_addr->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

first_discovered( )

Use this method to retrieve the date and time that the DHCP Fixed Address object was first discovered.

Parameter

none

Returns

The method returns the date and time that the DHCP Fixed Address object was first discovered.

Example

```
#Get first_discovered
my $first_discovered = $fixed_addr->first_discovered();
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the ignore_dhcp_option_list_request flag of a DHCP Fixed address object. If this flag is set to false all the defined DHCP options will be returned to the client.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate/unset it. The default value is false which indicates that this attribute inherit the member level setting..

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $fixed_addr->ignore_dhcp_option_list_request();
# Un-override ignore_dhcp_option_list_request
$fixed_addr->ignore_dhcp_option_list_request(undef);
# Modify ignore_dhcp_option_list_request
$fixed_addr->ignore_dhcp_option_list_request("true");
```

ipv4addr( )

Use this method to set or retrieve the ipv4addr value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $fixed_addr->ipv4addr();
#Modify ipv4addr
$fixed_addr->ipv4addr("10.0.0.5");
```

last_discovered( )

Use this method to retrieve the time this object was last seen by a discovery job.

The default value for this field is empty.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get last_discovered
my $last_discovered = $fixed_addr->last_discovered();
```

mac( )

Use this method to set or retrieve the MAC address value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

MAC Address of the DHCP Fixed address object. Format for the hexadecimal strings for MAC addresses and vendor prefixes with colons or dashes. Both of the following formats are acceptable: 11:11:11:11:11:11 and 11-11-11-11-11-11.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mac
my $mac = $fixed_addr->mac();
#Modify mac
$fixed_addr->mac("11::11::11::11::11::12");
```

match_client( )

Use this method to set or retrieve the match_client value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

```
    match_client "MAC": The fixed IP address is leased to the matching MAC address.

    match_client "CLIENT_IDENTIFIER": The fixed IP address is leased to the matching DHCP client
identifier

    match_client "RESERVED": The fixed IP address is reserved for later use with a MAC address that
only has zeros.

    match_client "CIRCUIT_ID": The fixed IP address is leased to the DHCP client with a matching
```

circuit ID. Note that the "agent_circuit_id" field must be set in this case.

match_client "REMOTE_ID": The fixed IP address is leased to the DHCP client with a matching remote ID. Note that the "agent_remote_id" field must be set in this case.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get match_client
my $match_client = $fixed_addr->match_client();
#Modify match_client
$fixed_addr->match_client("CLIENT_IDENTIFIER");
```

member( )

Use this method to set or retrieve the Microsoft server associated with this fixed address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a defined Infoblox::DHCP::MSServer object. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get member
my $ref_member = $dhcp_fixedaddr->member();
#Modify member
$dhcp_fixedaddr->member($member1); #$member1 is an Infoblox::DHCP::MSServer object
```

name( )

Use this method to set or retrieve the name of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A string specifying the name of the fixed address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $fixed_addr->name();
#Modify name
$fixed_addr->name("fxname");
```

netbios( )

Use this method to retrieve the name in the NetBIOS reply that responded to a NetBIOS query.

Returns

The method returns the attribute value.

Example

```
#Get netbios
my $netbios = $fixed_addr->netbios();
```

ms_options( )

Use this method to set or retrieve the Microsoft server custom DHCP options associated with this DHCP fixed address.

Note: You can set only options that are specified in both the Microsoft server and NIOS option definitions. Options that are not common to both the Microsoft server and NIOS cannot be modified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::MSOption objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $ref_options = $dhcp_fixedaddr->ms_options();
#Modify options
$dhcp_fixedaddr->ms_options([$option12]); #$option12 is an Infoblox::DHCP::MSOption object
```

network( )

Use this method to retrieve the network value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An network is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points followed by prefix (example: 1.0.0.0/8).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network
my $network = $fixed_addr->network();
```

network_component_description( )

Use this method to retrieve the description of the network device that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the network device description.

Example

```
#Get network_component_description
my $network_component_description = $fixed_addr->network_component_description();
```

network_component_ip( )

Use this method to retrieve the IP address of the network device that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the IP address of the network device.

Example

```
#Get network_component_ip
my $network_component_ip = $fixed_addr->network_component_ip();
```

network_component_name( )

Use this method to retrieve the name of the network device that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the network device name.

Example

```
#Get network_component_name
my $network_component_name = $fixed_addr->network_component_name();
```

network_component_port_description( )

Use this method to retrieve the description of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the description of the port on the network device.

Example

```
#Get network_component_port_description
my $network_component_port_description = $fixed_addr->network_component_port_description();
```

network_component_port_name( )

Use this method to retrieve the name of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the name of the port on the network device.

Example

```
#Get network_component_port_name
my $network_component_port_name = $fixed_addr->network_component_port_name();
```

network_component_port_number( )

Use this method to retrieve the number of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the network device port number in the range 0-9999.

Example

```
#Get network_component_port_number
my $network_component_port_number = $fixed_addr->network_component_port_number();
```

network_component_type( )

Use this method to retrieve the type of the network device that is connected to DHCP Fixed Address object.

Parameter

none

Returns

The method returns the network device type.

Example

```
#Get network_component_type
my $network_component_type = $fixed_addr->network_component_type();
```

network_view( )

Use this method to set or retrieve the network view of the DHCP fixed address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the fixed address is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $fixed_addr->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$fixed_addr->network_view($nview);
```

nextserver( )

Use this method to set or retrieve the nextserver value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::FixedAddr->bootfile() and Infoblox::DHCP::FixedAddr->bootserver().

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nextserver
my $nextserver = $fixed_addr->nextserver();
#Modify nextserver
$fixed_addr->nextserver("3.3.3.4");
#Un-override nextserver
$fixed_addr->nextserver(undef);
```

options( )

Use this method to set or retrieve the options of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

See Infoblox::DHCP::Option for parameters and methods available for this object.

Parameter

Options describe network configuration settings and various services available on the network. These options occur as variable-length fields at the end of DHCP messages.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $fixed_addr->options();
#Modify options
$fixed_addr->options([$option1, $option2]);
```

os( )

Use this method to retrieve the operating system associated with this fixed address.

Returns

The method returns the attribute value.

Example

```
#Get os
my $os = $fixed_addr->os();
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP Fixed Address is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$fixed_addr->override_ddns_domainname( );
#Modifying override_ddns_domainname
$fixed_addr->override_ddns_domainname("true");
```

port_duplex( )

Use this method to retrieve the duplex setting of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the port duplex setting. Returned values, if any, are one of the following:

```
Full
Half
```

Example

```
#Get port_duplex
my $port_duplex = $fixed_addr->port_duplex();
```

port_link_status( )

Use this method to retrieve the link status of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the network device port link status. Returned values, if any, are one of the following:

```
Connected
Not Connected
Unknown
```

Example

```
#Get port_link_status
my $port_link_status = $fixed_addr->port_link_status();
```

port_speed( )

Use this method to retrieve the speed of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the network device port speed. Returned values, if any, are one of the following:

```
10M
100M
1G
10G
100G
Unknown
```

Example

```
#Get port_speed
my $port_speed = $fixed_addr->port_speed();
```

port_status( )

Use this method to retrieve the status of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the network device port status. Returned values, if any, are one of the following:

```
Up
Down
Unknown
```

Example

```
#Get port_status
my $port_status = $fixed_addr->port_status();
```

port_vlan_description( )

Use this method to retrieve the VLAN description of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the VLAN description of the network device port.

Example

```
#Get port_vlan_description
my $port_vlan_description = $fixed_addr->port_vlan_description();
```

port_vlan_name( )

Use this method to retrieve the VLAN name of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the network device port VLAN name.

Example

```
#Get port_vlan_name
my $port_vlan_name = $fixed_addr->port_vlan_name();
```

port_vlan_number( )

Use this method to retrieve the VLAN number of the network device port that is connected to the DHCP Fixed Address object.

Parameter

none

Returns

The method returns the port VLAN number in the range 0-9999.

Example

```
#Get port_vlan_number
my $port_vlan_number = $fixed_addr->port_vlan_number();
```

pxe_lease_time( )

Use this method to set or retrieve the pxe_lease_time value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter appropriate values in the Days, Hours, Mins and Secs fields. User can specify the duration of time it takes a host to connect to a boot server, such as a TFTP server, and download the file it needs to boot.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get pxe_lease_time
my $pxe_lease_time = $fixed_addr->pxe_lease_time();
#Modify pxe_lease_time
$fixed_addr->pxe_lease_time("369");
```

template( )

Use this method to set or retrieve the fixed address template name for a DHCP Fixed Address object.

Template is the name of a DHCP Fixed Address template which will be used to create the DHCP Fixed Address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid name in sting format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

The template parameter can be used to only add a new DHCP Fixed Address and cannot be used during object modify or returned during object get.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get template
my $template = $fixed_addr->template();
#Modify template
$fixed_addr->template("Template_Name");
```

v_cluster( )

Use this method to retrieve the name of the VMware cluster associated with the DHCP Fixed Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware cluster name. The default value is an empty string.

Example

```
#Get v_cluster
my $v_cluster = $fixed_addr->v_cluster();
```

v_datacenter( )

Use this method to retrieve the name of the VMware datacenter associated with the DHCP Fixed Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware datacenter name. The default value is an empty string.

Example

```
#Get v_datacenter
my $v_datacenter = $fixed_addr->v_datacenter();
```

v_host( )

Use this method to retrieve the name of the VMware host associated with the DHCP Fixed Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware host name. The default value is an empty string.

Example

```
#Get v_host
my $v_host = $fixed_addr->v_host();
```

v_name( )

Use this method to retrieve the name of the VMware entity associated with the DHCP Fixed Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware entity name. The default value is an empty string.

Example

```
#Get v_name
my $v_name = $fixed_addr->v_name();
```

v_netadapter( )

Use this method to retrieve the name of the physical network adapter through which the VMware entity is connected to the appliance. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the network adapter name. The default value is an empty string.

Example

```
#Get v_netadapter
my $v_netadapter = $fixed_addr->v_netadapter();
```

v_switch( )

Use this method to retrieve the name of the virtual switch through which the VMware entity is connected to the appliance. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the virtual switch name. The default value is an empty string.

Example

```
#Get v_switch
my $v_switch = $fixed_addr->v_switch();
```

v_type( )

Use this method to retrieve the type of VMware entity associated with the DHCP Fixed Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware entity type. The default value is an empty string. The returned value, if any, can be one of the following:

```
VirtualMachine
HostSystem
VirtualCenter
```

Example

```
#Get v_type
my $v_type = $fixed_addr->v_type();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP Fixed Address object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
}
print "Session created successfully\n";

#Create a network object
my $network = Infoblox::DHCP::Network->new(
                                          "network" => "10.0.0.0/8",
                                         );
unless($network) {
     die("Construct Network object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

$session->add($network)
    or die("Add network failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Network added successfully\n";

my $option1 = Infoblox::DHCP::Option->new(
                                         "name"  => "lease-time",
                                         "value" => "50"
                                        );
unless($option1) {
     die("Construct option1 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option1 object created successfully\n";

my $option2 = Infoblox::DHCP::Option->new(
                                         "type"      => "routers",
                                         "ipv4addrs" => [ "10.0.0.1", "10.0.0.255" ],
                                        );
unless($option2) {
     die("Construct option2 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option2 object created successfully\n";
```

#Create a DHCP Fixed address object

```
#Create a DHCP Fixed Address object
my $fixed_addr = Infoblox::DHCP::FixedAddr->new(
    "ipv4addr"       => "10.0.0.6",
    "mac"            => "00:00:00:11:11:11",
    "network"        => "10.0.0.0/8",
    "comment"        => "add fixed address",
    "enable_ddns"    => "true",
    "pxe_lease_time" => "3600",
    "bootserver"     => "someserver.com",
    "nextserver"     => "3.3.3.3",
    "bootfile"       => "bootfile.com",
    "options"        => [ $option1 ,  $option2 ],
);
unless($fixed_addr) {
     die("Construct fixed address failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Fixed address object created successfully\n";

#Verify if the DHCP Fixed Address exists in the Infoblox appliance
my $object = $session->get(
       object => "Infoblox::DHCP::FixedAddr",
       mac    => "00:00:00:11:11:11"
);
unless ($object) {
   print "fixed address does not exist on server, safe to add the fixed address\n";
   $session->add($fixed_addr)
      or die("Add fixed address failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Fixed address added successfully\n";
```

#Create a DHCP Fixed Address object using a template

```
#Create a DHCP Fixed Address template object
my $fixed_addr_template = Infoblox::DHCP::FixedAddrTemplate->new(
```

```
    "name"                  => "custom_fa_template",
    "offset"                => "10",
    "number_of_addresses"    => "1",
    "comment"               => "add fixed address template",
    "enable_ddns"           => "TRUE",
);
unless($fixed_addr_template) {
     die("Construct fixed address failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Fixed address template object created successfully\n";

$session->add($fixed_addr_template)
      or die("Add fixed address template failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Fixed address template added successfully\n";

#Create a DHCP Fixed Address object
my $fixed_addr = Infoblox::DHCP::FixedAddr->new(
    "network"        => "10.0.0.0/8",
    "ipv4addr"       => "10.0.0.7",
    "match_client"   => "RESERVED",
    "template"       => "custom_fa_template"
);
unless($fixed_addr) {
     die("Construct fixed address failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Fixed address object created successfully\n";

$session->add($fixed_addr)
      or die("Add fixed address failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Fixed address added successfully\n";
```

#Search for a specific DHCP Fixed Address object

```
#Search for all DHCP Fixed address objects under the network 10.0.0.0/8
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::FixedAddr",
    mac     => "00:00:00:11:11.*",
    network => "10.0.0.0/8" );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search for Fixed Address object failed: ",
              $session->status_code() . ":" . $session->status_detail());
}
print "Search Fixed Address object found at least 1 matching entry\n";
```

#Get and modify a DHCP Fixed Address object

```
#Get DHCP Fixed Address object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FixedAddr",
    mac    => "00:00:00:11:11:11",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Fixed Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Fixed Address object found at least 1 matching entry\n";

#Modify one of the attributes of the obtained Fixed Address object
$object->pxe_lease_time("1234");

#Un-override bootfile
$object->bootfile(undef);

#Un-override bootserver
$object->bootserver(undef);

#Un-override nextserver
$object->nextserver(undef);
```

```
#Apply the changes
$session->modify($object)
    or die("Modify Fixed Address object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Fixed Address object modified successfully \n";
```

#Remove a Fixed Address object

```
#Get Fixed Address object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FixedAddr",
    mac    => "00:00:00:11:11:11",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Fixed Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Fixed Address object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove Fixed Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Fixed Address object removed successfully \n";

#Submit the Network object for removal
$session->remove($network)
    or die("Remove Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

#Submit the Fixed Address Template object for removal
$session->remove($fixed_addr_template)
    or die("Remove Fixed Address Template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Fixed Address Template object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::Network, Infoblox::DHCP::View, Infoblox::DHCP::Range, Infoblox::DHCP::Lease,Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session, Infoblox::DHCP::FixedAddrTemplate

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::FixedAddrTemplate - DHCP Fixed Address template object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::NetworkTemplate-> fixed_address_templates( )
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
- METHODS
  - bootfile( )
  - bootserver( )
  - comment( )
  - deny_bootp( )
  - enable_ddns( )
  - extensible_attributes( )
  - ddns_domainname( )
  - ddns_hostname( )
  - ignore_dhcp_option_list_request( )
  - name( )
  - nextserver( )
  - number_of_addresses( )
  - offset( )
  - options( )
  - override_ddns_domainname( )
  - pxe_lease_time( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::FixedAddrTemplate - DHCP Fixed Address template object.

DESCRIPTION

A fixed address template works as a 'cookie cutter' and can be used to create a real fixed address object in a quick and consistant way. A Fixed address object created from a fixed address template will inherit most properties defined in fixed address template object so most of the fixed address template properties are the same as the fixed address object properties.

CONSTRUCTOR

```
 my $fixed_addr_template = Infoblox::DHCP::FixedAddrTemplate -> new(
       name                              => $string,                      #Required
       number_of_addresses               => $num | undef,                 #Optional / Default is undefined
       offset                            => $num | undef,                 #Optional / Default is undefined
       bootfile                          => $filename | undef,            #Optional / Default is undefined
       bootserver                        => $ipv4addr | $fqdn | undef,    #Optional / Default is undefined
       comment                           => $string,                      #Optional / Default is empty
       deny_bootp                        => "true" | "false" | undef,     #Optional / Default is undefined
       enable_ddns                       => "true" | "false",             #Optional / Default is false
       ddns_domainname                   => $string | undef,              #Optional / Default is undefined
       ddns_hostname                     => $string | undef,              #Optional / Default is undefined
       extensible_attributes             => { $string => $string | $num, $string => [ $string | $num, ...
], ... } #Optional / Default is undefined

       ignore_dhcp_option_list_request   => "true" | "false" | undef,     #Optional / Default is false
       nextserver                        => $ipv4addr | $fqdn | undef,    #Optional / Default is undefined
       options                           => [$option1, $option2, ...]     #Optional / Default is empty
       override_ddns_domainname          => "true" | "false",             #Optional / Default value is
"false"
       pxe_lease_time                    => $time,                        #Optional / Default is empty
        );
```

MODULE METHODS

The following functions are available to be applied to a DHCP fixed address template object.

Infoblox::DHCP::NetworkTemplate-> fixed_address_templates( )

Use this function to specify a fixed address template at network template on the Infoblox appliance. See Infoblox::DHCP::NetworkTemplate->fixed_address_templates() for parameters and return values.

Example

```
#Create the Network Template object with fixed address template
 my $template = Infoblox::DHCP::Template->new(
   name                    => "custom_fa_template"
 );
 my $network_template = Infoblox::DHCP::NetworkTemplate->new(
   name                    => "custom_network_template",
   comment                 => "add network",
   fixed_address_templates => [$template]
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that can be applied to a DHCP Fixed Address template object.

Infoblox::Session->add( )

Use this method to add a DHCP Fixed Address template object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a DHCP Fixed Address template object
my $fixed_addr_template = Infoblox::DHCP::FixedAddrTemplate->new(
      "name"                 => "custom_fa_template",
      "offset"               => "10",
      "number_of_addresses"   => "1",
      "bootfile"             => "bootfile",
      "bootserver"           => "bootserver.com",
      "comment"              => "add fixed address",
      "deny_bootp"           => "false",
      "enable_ddns"          => "false",
      "nextserver"           => "3.3.3.3",
      "options"              => [ $option1,  $option2 ],
      "pxe_lease_time"        => "3600",
);

#Submit for addition
my $response = $session->add($fixed_addr_template);
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP Fixed Address template object:

name - Optional. The name of the Fixed address template.
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
#Get DHCP Fixed Address template object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FixedAddrTemplate",
    name   => "custom_fa_template",
);

my @retrieved_objs = $session->get(
  object                => "Infoblox::DHCP::FixedAddrTemplate",
  extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->search( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP Fixed Address template object:

name    - Optional. The name of the fixed address template (regular expression).
comment - Optional. The comment of the fixed address template (regular expression).
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
#Search DHCP Fixed Address template object through the session
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::FixedAddrTemplate",
    name    => "custom_fa_template",
    comment => ".*",
);

#Search for DHCP Fixed Address template objects defining "Santa Clara" for "Site" extensible attribute
my @retrieved_objs = $session->search(
    object                => "Infoblox::DHCP::FixedAddrTemplate",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP Fixed Address template object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the pxe_lease_time
$fixed_addr_templateess->pxe_lease_time("1234");
#Submit modification
my $response = $session->modify( $fixed_addr_templateess );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP Fixed Address template object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` to retrieve the specific DHCP Fixed Address template object, and then submit this object for removal.

Example

```
#Get DHCP Fixed Address template object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FixedAddrTemplate",
    name   => "custom_fa_template",
);
#Find the desired object from the retrieved list.
my $desired_fixed_addr_template = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_fixed_addr_template );
```

METHODS

This section describes all the methods that can be used to set and retrieve the attribute values of a DHCP Fixed Address template object.

bootfile( )

Use this method to set or retrieve the bootfile value of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::FixedAddrTemplate->bootserver() and Infoblox::DHCP::FixedAddrTemplate->nextserver().

Parameter

The name of the file that the client must download. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootfile
my $bootfile = $fixed_addr_template->bootfile();
#Modify bootfile
$fixed_addr_template->bootfile("boot_file");
#Un-override bootfile
$fixed_addr_template->bootfile(undef);
```

bootserver( )

Use this method to set or retrieve the bootserver of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::FixedAddrTemplate->bootfile() and Infoblox::DHCP::FixedAddrTemplate->nextserver().

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootserver
my $bootserver = $fixed_addr_template->bootserver();
#Modify bootserver
$fixed_addr_template->bootserver("bootserver2.com");
#Un-override bootserver
$fixed_addr_template->bootserver(undef);
```

comment( )

Use this method to set or retrieve the descriptive comment of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $fixed_addr_template->comment();
#Modify comment
$fixed_addr_template->comment("Modified DHCP Fixed Address template object comment");
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp flag of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_bootp flag or "false" to deactivate/unset it. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $fixed_addr_template->deny_bootp();
#Modify deny_bootp
$fixed_addr_template->deny_bootp("true");
```

enable_ddns( )

Use this method to set or retrieve the enable_ddns flag of a DHCP Fixed Address template object. If enable_ddns value is specified as "true", then the A and PTR records corresponding to the Fixed Address template object will be placed on the DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_ddns flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ddns
my $enable_ddns = $fixed_addr_template->enable_ddns();
#Modify enable_ddns
$fixed_addr_template->enable_ddns("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP Fixed Address template object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $fixed_addr_template->extensible_attributes();
#Modify extensible attributes
$fixed_addr_template->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [
'Peter', 'Tom' ] });
```

ddns_domainname( )

Use this method to set or retrieve the ddns_domainname value of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_dddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

Desired name for the dynamic DNS domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_domainname
my $ddns_domainname = $fixed_addr->ddns_domainname();
#Modify ddns_domainname
$fixed_addr->ddns_domainname("test_domain.com");
```

ddns_hostname( )

Use this method to set or retrieve the ddns_hostname value of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name for the dynamic DNS host name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_hostname
my $ddns_hostname = $fixed_addr->ddns_hostname();
#Modify ddns_hostname
$fixed_addr->ddns_hostname("test_host");
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the ignore_dhcp_option_list_request flag of a DHCP fixed address teplate object. If this flag is set to false all the defined DHCP options will be returned to the client. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate it. The default value is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $fixed_addr_template->ignore_dhcp_option_list_request();
# Modify ignore_dhcp_option_list_request
$fixed_addr_template->ignore_dhcp_option_list_request("true");
# Un-override ignore_dhcp_option_list_request
$fixed_addr_template->ignore_dhcp_option_list_request(undef);
```

name( )

Use this method to set or retrieve the name of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the fixed address template in string format with a maximum of 64 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.
If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $fixed_address_template->name();
#Modify name
$fixed_address_template->name("custom_fa_template");
```

nextserver( )

Use this method to set or retrieve the nextserver value of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::FixedAddrTemplate->bootfile() and Infoblox::DHCP::FixedAddrTemplate->bootserver().

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nextserver
my $nextserver = $fixed_addr_template->nextserver();
#Modify nextserver
$fixed_addr_template->nextserver("3.3.3.4");
#Un-override nextserver
$fixed_addr_template->nextserver(undef);
```

number_of_addresses( )

Use this method to set or retrieve the number of addresses in a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of addresses in this fixed address template.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get number_of_addresses
my $number_of_addresses = $fixed_address_template-> number_of_addresses();
#Modify number_of_addresses
$fixed_address_template->number_of_addresses("10");
```

offset( )

Use this method to set or retrieve the address offset of a Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The address offset of this fixed address template. The offset is calculated from the start address of network template to which the fixed address template is assigned to. For example, if offset is set to 10 and number_of_addresses is set to 10, and network is set to 10.0.0.0/8, then there will be 10 fixed address object created and start address will be 10.0.0.10 and end address will be 10.0.0.20.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get offset
```

```
my $offset = $fixed_address_template->offset();
#Modify offset
$fixed_address_template->offset("10");
```

options( )

Use this method to set or retrieve the options of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

See Infoblox::DHCP::Option for parameters and methods available for this object.

Parameter

Options describe network configuration settings and various services available on the network. These options occur as variable-length fields at the end of DHCP messages.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $fixed_addr_template->options();
#Modify options
$fixed_addr_template->options([$option1, $option2]);
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP Fixed Address is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$fixed_addr_template->override_ddns_domainname( );
#Modifying override_ddns_domainname
$fixed_addr_template->override_ddns_domainname("true");
```

pxe_lease_time( )

Use this method to set or retrieve the pxe_lease_time value of a DHCP Fixed Address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter appropriate values in the Days, Hours, Mins and Secs fields. User can specify the duration of time it takes a host to connect to a boot server, such as a TFTP server, and download the file it needs to boot.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get pxe_lease_time
my $pxe_lease_time   $fixed_addr_template   pxe_lease_time
```

```
                                =                        ->              ();
        #Modify pxe_lease_time
        $fixed_addr_template->pxe_lease_time("369");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, get, modify, and remove. Also, this sample includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $option1 = Infoblox::DHCP::Option->new(
                                          "name"  => "fqdn",
                                          "value" => "mydomain.com"
                                         );
unless($option1) {
     die("Construct option1 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option1 object created successfully\n";

my $option2 = Infoblox::DHCP::Option->new(
                                          "name"     => "merit-dump",
                                           "value"   => "sometextdump",
                                         );
unless($option2) {
     die("Construct option2 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option2 object created successfully\n";
```

#Create a DHCP Fixed Address template object

```
#Create a DHCP Fixed Address template object
my $fixed_addr_template = Infoblox::DHCP::FixedAddrTemplate->new(
    "name"                  => "custom_fa_template",
    "offset"                => "10",
    "number_of_addresses"    => "1",
    "comment"               => "add fixed address template",
    "enable_ddns"           => "TRUE",
    "ddns_domainname"        => "test_domain.com",
    "ddns_hostname"          => "test_host",
    "pxe_lease_time"         => "3600",
    "bootserver"            => "someserver.com",
    "nextserver"            => "3.3.3.3",
    "bootfile"              => "bootfile.com",
    "options"               => [ $option1, $option2 ],
);
unless($fixed_addr_template) {
     die("Construct Fixed Address template failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Fixed Address template object created successfully\n";

#Verify if the DHCP Fixed Address templateexists in the Infoblox appliance
my $object = $session->get(
       object => "Infoblox::DHCP::FixedAddrTemplate",
       name   => "custom_fa_template"
);
```

```
unless ($object) {
   print "Fixed Address template does not exist on server, safe to add the fixed address template\n";
   $session->add($fixed_addr_template)
      or die("Add Fixed Address template failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Fixed Address template added successfully\n";
```

#Search a DHCP Fixed Address template object

```
#Search DHCP Fixed Address template object through the session
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::FixedAddrTemplate",
    name   => "custom_fa_template",
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Search Fixed Address template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search Fixed Address template object found at least 1 matching entry\n";
```

#Get and modify a DHCP Fixed Address template object

```
#Get DHCP Fixed Address template object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FixedAddrTemplate",
    name   => "custom_fa_template",
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get Fixed Address template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Fixed Address template object found at least 1 matching entry\n";

#Modify one of the attributes of the obtained Fixed Address template object
$object->pxe_lease_time("1234");

#Un-override bootfile
$object->bootfile(undef);

#Un-override bootserver
$object->bootserver(undef);

#Un-override nextserver
$object->nextserver(undef);

#Apply the changes
$session->modify($object)
  or die("Modify Fixed Address template failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "DHCP Fixed Address template added successfully\n";
```

#Create the Network Template object with fixed address template

```
my $template = Infoblox::DHCP::Template->new(
    name => "custom_fa_template",
);

my $network_template = Infoblox::DHCP::NetworkTemplate->new(
    name    => "custom_network_template",
    comment => "add network",
    fixed_address_templates => [$template]
);

unless($network_template) {
       die("Construct Network template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}

print "Network template object created successfully\n";

#Add the Network template object into the Infoblox appliance through a session
```

```
my $resp = $session->add($network_template)
or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "Network template object added to Infoblox appliance successfully\n";
```

#Remove a Fixed Address template object

```
#Remove a Network Template object
$session->remove($network_template)
    or die("Remove Network Template template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Network Template object removed successfully \n";

#Get Fixed Address template object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::FixedAddrTemplate",
    name   => "custom_fa_template",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Fixed Address template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Fixed Address template object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove Fixed Address template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Fixed Address template object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify(), Infoblox::DHCP::NetworkTemplate, Infoblox::DHCP::RangeTemplate,Infoblox::DHCP::Template,Infoblox::Session, Infoblox::DHCP::Option

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::IPv6FixedAddr - DHCP IPv6 Fixed Address object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - address_type( )
  - comment( )
  - configure_for_dhcp( )
  - disable( )
  - domain_name( )
  - domain_name_servers( )
  - duid( )
  - extensible_attributes( )
  - ipv6addr( )
  - ipv6prefix( )
  - ipv6prefix_bits( )
  - match_client( )
  - name( )
  - network( )
  - network_view( )
  - options( )
  - override_domain_name( )
  - override_domain_name_servers( )
  - override_options( )
  - override_preferred_lifetime( )
  - override_valid_lifetime( )
  - preferred_lifetime( )
  - template( )
  - valid_lifetime( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6FixedAddr - DHCP IPv6 Fixed Address object.

DESCRIPTION

An IPv6 Fixed Address is a specific IP address that a DHCP server always assigns when a lease request comes from a particular MAC address of the client. This is an IPv6 representation of a fixed address.

CONSTRUCTOR

```
 my $fixed_addr = Infoblox::DHCP::IPv6FixedAddr -> new(
       duid                            => $string,                     # Required
       address_type                    => "ADDRESS" | "PREFIX" | "BOTH", # Optional / Default is "ADDRESS"
       comment                         => $string,                     # Optional / Default is empty
       configure_for_dhcp              => "true" | "false",            # Optional / Default is "false"
                                                                       # When both the IP address and DUID
are present, the default value is "true"
                                                                       # Applicable only when adding this
fixed address to a host
       disable                         => "true" | "false",            # Optional / Default is "false"
       domain_name                     => $string,                     # Optional / Default is undefined
       domain_name_servers             => "$ipv6addr1,$ipv6addr2,...", # Optional / Default is undefined
       extensible_attributes           => { $string => $string | $num, $string => [ $string | $num, ... ],
... } #Optional / Default is undefined
       ipv6addr                        => $ipv6addr,                   # Optional / one of ipv6addr or
ipv6prefix is required
       ipv6prefix                      => $ipv6addr,                   # Optional / one of ipv6addr or
ipv6prefix is required
```

```
        ipv6prefix_bits                 => $ipv6addr,                   # Optional / Required if ipv6prefix
is specified
        match_client                    => $string,                     # Optional / Default is "DUID"
                                                                       # Possible values are:
                                                                       #  "DUID", "RESERVED",
        name                            => $string,                     # Optional / Default is undefined
        network                         => $ipv6network,                # Optional / Default is undefined
        network_view                    => $NetworkView,                # Optional / Default is "default"
network view
        options                         => [$option1, $option2, ...]    # Optional / Default is empty
        override_domain_name            => "true" | "false",            # Optional / Default value is
"false"
        override_domain_name_servers    => "true" | "false",            # Optional / Default value is
"false"
        override_options                => "true" | "false",            # Optional / Default value is
"false"
        override_preferred_lifetime     => "true" | "false",            # Optional / Default value is
"false"
        override_valid_lifetime         => "true" | "false",            # Optional / Default value is
"false"
        preferred_lifetime              => $num,                        # Optional / Default is "27000"
        template                        => $string,                     # Optional / Default is undefined
        valid_lifetime                  => $num,                        # Optional / Default is 43200
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DHCP IPv6 Fixed Address object.

Infoblox::Session->add( )

Use this method to add a DHCP IPv6 Fixed Address object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a DHCP IPv6 Fixed Address object
my $ipv6fa = Infoblox::DHCP::IPv6FixedAddr->new(
        duid                    => "0001",
        comment                 => "add ipv6 fixed address using uid",
        disable                 => "false",
        match_client            => "DUID",
        network                 => "2001::/24",
        network_view            => $nview,
        options                 => [ $option1,  $option2, $option3,  $option4, $option5, $option6 ],
);

#Submit for addition
my $response = $session->add($ipv6fa)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 Fixed Address object:

 comment                 - Optional. User-specified comment for the fixed address.
 duid                    - Optional. DUID of the fixed address.
 ipv6addr                - Optional. IPv6 address of the fixed address
 network                 - Optional. Network address of the fixed address.
 network_view            - Optional. Network view in which the fixed address is located. By default,
all network views are searched.
 extensible_attributes   - Optional. A hash reference containing extensible attributes.
```

Examples

```
#Get DHCP IPv6 Fixed Address object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6FixedAddr",
    duid   => "0001",
);
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::IPv6FixedAddr",
    ipv6addr     => "2001::1",
    network_view => "default",
```

```
);

my @retrieved_objs = $session->get(
   object                => "Infoblox::DHCP::IPv6FixedAddr",
   extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP IPv6 Fixed Address object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the preferred_lifetime
$ipv6fa->preferred_lifetime("1234567");
#Submit modification
my $response = $session->modify( $ipv6fa );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP IPv6 Fixed Address object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP IPv6 Fixed Address object, and then submit this object for removal.

Example

```
#Get DHCP IPv6 Fixed Address object through the session
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::IPv6FixedAddr",
    duid         => "0001",
    network_view => "default"
);
#Find the desired object from the retrieved list.
my $desired_ipv6fa = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_ipv6fa );
```

Infoblox::Session->search( )

Use this method to search for DHCP IPv6 Fixed Address object and matching Host object with MAC address in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP IPv6 Fixed Address object:

 comment                 - Optional. User-specified comment for the fixed address.
 duid                    - Optional. DUID of the fixed address.
 ipv6addr                - Optional. IPv6 address of the fixed address
 network                 - Optional. Network address of the fixed address.
 network_view            - Optional. Network view in which the fixed address is located. By default,
all network views are searched.
 extensible_attributes   - Optional. A hash reference containing extensible attributes.
```

Examples

```
# Search for all DHCP IPv6 Fixed Address objects that are specified in the network "2001::/24" in
the "default" network view.
my @retrieved_objs = $session->search(
    object       => "Infoblox::DHCP::IPv6FixedAddr",
    duid         => "0001",
    network      => "2001::/24",
    network_view => "default"
);

# Search for all DHCP IPv6 Fixed Address objects with "Santa Clara" for the "Site" extensible
attribute
my @retrieved_objs = $session->search(
    object                => "Infoblox::DHCP::IPv6FixedAddr",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a DHCP IPv6 Fixed Address object.

address_type( )

Use this method to set or retrieve the address type of a DHCP IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "ADDRESS", "PREFIX", or "BOTH". When the address type is "ADDRESS", a value for the 'ipv6addr' member is required. When the address type is "PREFIX", values for 'ipv6prefix' and 'ipv6prefix_bits' are required. When the address type is "BOTH", values for 'ipv6addr', 'ipv6prefix', and 'ipv6prefix_bits' are all required.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get address_type
my $address_type = $ipv6fa->address_type();
#Modify address_type
$ipv6fa->address_type("PREFIX");
```

comment( )

Use this method to set or retrieve the descriptive comment of a DHCP IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $ipv6fa->comment();
#Modify comment
$ipv6fa->comment("Modified DHCP IPv6 Fixed Address object comment");
```

configure_for_dhcp( )

Use this method to set or retrieve the configure_for_dhcp flag of a DHCP IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the configure_for_dhcp flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get configure_for_dhcp
my $configure_for_dhcp = $ipv6fa->configure_for_dhcp();
#Modify configure_for_dhcp
$ipv6fa->configure_for_dhcp("true");
```

disable( )

Use this method to set or retrieve the disable flag of a DHCP IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $fixed_addr->disable();
#Modify disable
$fixed_addr->disable("true");
```

domain_name( )

Use this method to set or retrieve the domain_name value of a DHCP IPv6 fixed address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_domain_name attribute to "false".

Note that when domain_name is set to a defined value and override_domain_name is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name("testdomain"); $object->override_domain_name("false"); will set override_domain_name to "false", and the sequence $object->override_domain_name("false"); $object->domain_name("testdomain"); will result in override_domain_name="true".

Parameter

The client's domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name
my $domain_name = $fixed_addr->domain_name();
#Modify domain_name
$fixed_addr->domain_name("test_domain.com");
```

domain_name_servers( )

Use this method to set or retrieve the domain_name_servers value of a DHCP IPv6 fixed address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name_servers method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_domain_name_servers attribute to "false".

Note that when domain_name_servers is set to a defined value and override_domain_name_servers is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name_servers("2001::1,2001::2"); $object->override_domain_name_servers("false"); will set override_domain_name_servers to "false", and the sequence $object->override_domain_name_servers("false"); $object->domain_name_servers("2001::1,2001::2"); will result in override_domain_name_servers="true".

Parameter

The comma-separated list of domain name server addresses in IPv6 address format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name_servers
my $domain_name_servers = $fixed_addr->domain_name_servers();
#Modify domain_name_servers
$fixed_addr->domain_name_servers("2001::1,2001::2");
```

duid( )

Use this method to set or retrieve the DHCPv6 Unique Identifier (DUID) of the IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DUID of the matching client in hexadecimal format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get duid
my $duid = $ipv6fa->duid();
#Modify duid
$ipv6fa->duid("1c2edfaa");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with an IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $ipv6fa->extensible_attributes();
#Modify extensible attributes
$ipv6fa->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

ipv6addr( )

Use this method to set or retrieve the the IPv6 address of the IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv6 address is a 128-bit number represented as eight groups of four hexadecimal digits (i.e. two octets), each group separated by a `colon(:)` (example: 2001:0db8:85a3:0000:0000:8a2e:0370:7334). Leading zeroes and groups of consecutive zeros within the value can be omitted to produce a more compact representation (example: 2001::1, ::1), and dotted-quad notation can be used to express IPv4-mapped IPv6 addresses (example: ::ffff:192.0.2.128).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6addr
my ipv6addr = $ipv6fa->ipv6addr();
#Modify ipv6addr
$ipv6fa->ipv6addr("aaaa:1111:bbbb:2222:cccc:3333:dddd:5555");
```

ipv6prefix( )

Use this method to set or retrieve the IPv6 prefix of a DHCP IPv6 Fixed Address object. The prefix must belong to the list of global prefixes defined in the Infoblox::Grid::DHCP object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

IPv6 prefix from the list of global IPv6 prefixes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. Is.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6prefix
my $ipv6prefix = $ipv6fa->ipv6prefix();
#Modify ipv6prefix
$ipv6fa->ipv6prefix("2001::");
```

ipv6prefix_bits( )

Use this method to set or retrieve the prefix bits for the DHCP IPv6 Fixed Address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 2 and 128.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6prefix_bits
my $ipv6prefix_bits = $ipv6fa->ipv6prefix_bits();
#Modify ipv6prefix_bits
$ipv6fa->ipv6prefix_bits(32);
```

match_client( )

Use this method to set or retrieve the match_client value of a DHCP Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

```
match_client "DUID": The fixed IP address is leased to the matching DUID.
```

Returns

```
If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.
```

Example

```
#Get match_client
my $match_client = $ipv6fa->match_client();
#Modify match_client
$ipv6fa->match_client("DUID");
```

name( )

Use this method to set or retrieve the name of a DHCP IPv6 Fixed Address object. The same name can be used for multiple fixed addresses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify a name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the name
my $name = $ipv6fa->name();
#Modify the name
$ipv6fa->name("pop-server");
```

network( )

Use this method to retrieve the network value of a DHCP IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the address and netmask of the network in CIDR format (example: 2001::/64).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network
my $network = $ipv6addr->network();
#Modify network
$ipv6fa->network("2001::/24");
```

network_view( )

Use this method to set or retrieve the network view of the DHCP fixed address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the fixed address is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $ipv6fa->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$ipv6fa->network_view($nview);
```

options( )

Use this method to set or retrieve the options of a DHCP IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_options method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_options attribute to "false".

Note that when the options method is set to a defined value and override_options is set to "false", the last operation takes precedence. Thus the sequence $object->options([$opt]); $object->override_options("false"); will set override_options to "false", and the sequence $object->override_options("false"); $object->options([$opt]); will result in override_options="true".

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $ipv6fa->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name   => "dhcp6.rapid-commit",
    value  => "true"
);
$ipv6fa->options([$option1);
```

override_domain_name( )

The override_domain_name attribute controls whether the domain_name value in the DHCP IPv6 fixed address is used, instead of the upper-level default.

The override_domain_name attribute can be specified explicitly. It is also set implicitly when domain_name is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for domain_name. Set the parameter to "false" to inherit the upper-level setting for domain_name.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_domain_name
my $override_domain_name=$ipv6fa->override_domain_name( );
#Modify override_domain_name
$ipv6fa->override_domain_name("true");
```

override_domain_name_servers( )

The override_domain_name_servers attribute controls whether the domain_name_servers value in the DHCP IPv6 fixed address is used, instead of the upper-level default.

The override_domain_name_servers attribute can be specified explicitly. It is also set implicitly when domain_name_servers is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for domain_name_servers. Set the parameter to "false" to inherit the upper-level setting for domain_name_servers.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_domain_name_servers
my $override_domain_name_servers=$ipv6fa->override_domain_name_servers( );
#Modify override_domain_name_servers
$ipv6fa->override_domain_name_servers("true");
```

override_options( )

The override_options attribute controls whether the options in the DHCP IPv6 fixed address are used, instead of the upper-level default.

The override_options attribute can be specified explicitly. It is also set implicitly when options are set to defined values.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for valid_lifetime. Set the parameter to "false" to inherit the upper-level setting for valid_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_options
my $override_options=$fixed_addr->override_options( );
#Modify override_options
$fixed_addr->override_options("true");
```

override_preferred_lifetime( )

The override_preferred_lifetime attribute controls whether the preferred_lifetime value in the DHCP IPv6 fixed address is used, instead of the upper-level default.

The override_preferred_lifetime attribute can be specified explicitly. It is also set implicitly when preferred_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for valid_lifetime. Set the parameter to "false" to inherit the upper-level setting for valid_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_preferred_lifetime
my $override_preferred_lifetime=$fixed_addr->override_preferred_lifetime( );
#Modify override_preferred_lifetime
$fixed_addr->override_preferred_lifetime("true");
```

override_valid_lifetime( )

The override_valid_lifetime attribute controls whether the valid_lifetime value in the DHCP IPv6 fixed address is used, instead of the upper-level default.

The override_valid_lifetime attribute can be specified explicitly. It is also set implicitly when valid_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for valid_lifetime. Set the parameter to "false" to inherit the upper-level setting for valid_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_valid_lifetime
my $override_valid_lifetime=$fixed_addr->override_valid_lifetime( );
#Modify override_valid_lifetime
$fixed_addr->override_valid_lifetime("true");
```

preferred_lifetime( )

Use this method to set or retrieve the preferred_lifetime value of a DHCP IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_preferred_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_preferred_lifetime attribute to "false".

Note that when preferred_lifetime is set to a defined value and override_preferred_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->preferred_lifetime(20000); $object->override_preferred_lifetime("false"); will set override_preferred_lifetime to "false", and the sequence $object->override_preferred_lifetime("false"); $object->preferred_lifetime(20000); will result in override_preferred_lifetime="true".

Parameter

A 32-bit integer (valid range from 1 to 4294967295) that represents the lifetime, in seconds. The default value is 27000.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get preferred_lifetime
my $preferred_lifetime = $ipv6fa->preferred_lifetime();
#Modify preferred_lifetime
$ipv6fa->preferred_lifetime("369");
```

template( )

Use this method to set or retrieve the fixed address template name for a DHCP IPv6 Fixed Address object.

Template is the name of a DHCP IPv6 Fixed Address template which will be used to create the DHCP IPv6 Fixed Address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

The template parameter can be used to only add a new DHCP IPv6 Fixed Address and cannot be used during object modify or returned during object get.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get template
my $template = $ipv6fa->template();
#Modify template
$ipv6fa->template("Template_Name");
```

valid_lifetime( )

Use this method to set or retrieve the valid_lifetime value of a DHCP IPv6 Fixed Address object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer, from 1 to 4294967295, that represents the lifetime, in seconds. The default value is 43200.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get valid_lifetime
my $valid_lifetime = $ipv6fa->valid_lifetime();
#Modify valid_lifetime
$ipv6fa->valid_lifetime("36900");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP IPv6 Fixed Address object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create an IPv6 network object
my $network = Infoblox::DHCP::IPv6Network->new(
                                          "network" => "2001::/64",
                                         );
unless($network) {
     die("Construct IPv6 Network object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "IPv6 Network object created successfully\n";

$session->add($network)
    or die("Add network failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Network added successfully\n";

my $option1 = Infoblox::DHCP::Option->new(
    "name"  => "dhcp6.fqdn",
    "value" => "mydomain.com"
);
unless($option1) {
    die("Construct option1 object failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option1 object created successfully\n";

my $option2 = Infoblox::DHCP::Option->new(
    "name"     => "dhcp6.info-refresh-time",
    "value"    => "1000",
);
unless($option2) {
    die("Construct option2 object failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option2 object created successfully\n";
```

#Create a DHCP IPv6 Fixed address object

```
#Create a DHCP IPv6 Fixed Address object
my $fixed_addr = Infoblox::DHCP::IPv6FixedAddr->new(
```

```
        "ipv6addr"       => "2001::4",
        "network"        => "2001::/64",
        "duid"           => "000000111111",
        "comment"        => "add fixed address",
        "domain_name"    => "foo.com",
        "domain_name_servers" => "2002::1,2002::2",
        "preferred_lifetime" => 1234,
        "override_preferred_lifetime" => "true",
        "valid_lifetime" => 123456,
        "override_valid_lifetime" => "true",
        "options"        => [ $option1 , $option2 ],
        );
unless($fixed_addr) {
     die("Construct IPv6 fixed address failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6 Fixed address object created successfully\n";

#Verify if the DHCP IPv6 Fixed Address exists in the Infoblox appliance
my $object = $session->get(object => "Infoblox::DHCP::IPv6FixedAddr", duid => "00:00:00:11:11:11");
unless ($object) {
   print "IPv6 fixed address does not exist on server, safe to add the fixed address\n";
   $session->add($fixed_addr)
      or die("Add IPv6 fixed address failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "DHCP IPv6 Fixed address added successfully\n";
```

#Create a DHCP IPv6 Fixed address object using a template

```
#Create a DHCP IPv6 Fixed Address template object
my $fixed_addr_template = Infoblox::DHCP::IPv6FixedAddrTemplate->new(
    "name"                  => "custom_fa_template",
    "offset"                => "10",
    "number_of_addresses"   => "1",
    "comment"               => "add fixed address template",
    "valid_lifetime"        => 86400,
);
unless($fixed_addr_template) {
     die("Construct IPv6 fixed address failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6 Fixed address template object created successfully\n";

$session->add($fixed_addr_template)
      or die("Add IPv6 fixed address template failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Fixed address template added successfully\n";

#Create a DHCP IPv6 Fixed Address object
my $fixed_addr = Infoblox::DHCP::IPv6FixedAddr->new(
    "network"        => "2001::/64",
    "ipv6addr"       => "2001::2",
    "match_client"   => "DUID",
    "duid"           => "010203040505",
    "template"       => "custom_fa_template"
);
unless($fixed_addr) {
     die("Construct fixed address failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6 Fixed address object created successfully\n";

$session->add($fixed_addr)
      or die("Add fixed address failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Fixed address added successfully\n";
```

#Search for a specific DHCP IPv6 Fixed Address object

```
#Search for all DHCP IPv6 Fixed address objects under the network 2001::/64
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::IPv6FixedAddr",
    duid     => "0000001111.*",
    network => "2001::/64" );
my $object = $retrieved_objs[0];
```

```
unless ($object) {
       die("Search for IPv6 Fixed Address object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search IPv6 Fixed Address object found at least 1 matching entry\n";
```

#Get and modify a DHCP IPv6 Fixed Address object

```
#Get DHCP IPv6 Fixed Address object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6FixedAddr",
    duid    => "000000111111",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get IPv6 Fixed Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get IPv6 Fixed Address object found at least 1 matching entry\n";

#Modify one of the attributes of the obtained IPv6 Fixed Address object
$object->preferred_lifetime("1235");

#Apply the changes
$session->modify($object)
    or die("Modify IPv6 Fixed Address object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "IPv6 Fixed Address object modified successfully \n";
```

#Remove an IPv6 Fixed Address object

```
#Get IPv6 Fixed Address object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6FixedAddr",
    duid    => "000000111111",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get IPv6 Fixed Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get IPv6 Fixed Address object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove IPv6 Fixed Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "IPv6 Fixed Address object removed successfully \n";

#Submit the Network object for removal
$session->remove($network)
    or die("Remove Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

#Submit the Fixed Address Template object for removal
$session->remove($fixed_addr_template)
    or die("Remove IPv6 Fixed Address Template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "IPv6 Fixed Address Template object removed successfully \n";
####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::IPv6Network, Infoblox::DHCP::View, Infoblox::DHCP::IPv6Range, Infoblox::DHCP::IPv6Lease, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session, Infoblox::DHCP::IPv6FixedAddrTemplate

---

COPYRIGHT

Copyright (c) 2010 Infoblox Inc.

Infoblox::DHCP::IPv6FixedAddrTemplate - DHCP IPv6 Fixed Address Template object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::NetworkTemplate-> fixed_address_templates( )
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
- METHODS
  - comment( )
  - domain_name( )
  - domain_name_servers( )
  - extensible_attributes( )
  - name( )
  - number_of_addresses( )
  - offset( )
  - options( )
  - override_domain_name( )
  - override_domain_name_servers( )
  - override_options( )
  - override_preferred_lifetime( )
  - override_valid_lifetime( )
  - preferred_lifetime( )
  - valid_lifetime( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6FixedAddrTemplate - DHCP IPv6 Fixed Address Template object

DESCRIPTION

An IPv6 fixed address template contains a set of predefined properties and can be used to create an IPv6 fixed address object in a quick and consistent way. An IPv6 fixed address created from an IPv6 fixed address template inherits the properties defined in the template.

CONSTRUCTOR

```
my $fixed_addr_template = Infoblox::DHCP::IPv6FixedAddrTemplate -> new(
        name                        => $string,                              #Required
        comment                     => $string,                              #Optional / Default is
undefined
        domain_name                 => $fqdn,                                #Optional / Default is
undefined
        domain_name_servers         => "$ipv6addr1,$ipv6addr2,...",          #Optional / Default is
undefined
        extensible_attributes       => { $string => $string | $num, $string => [ $string | $num, ... ],
... }, #Optional / Default is undefined
        number_of_addresses         => $num | undef,                         #Optional / Default is
undefined
        offset                      => $num | undef,                         #Optional / Default is
undefined
        options                     => [$option1, $option2],                 #Optional / Default is empty
        override_domain_name        => "true" | "false",                     #Optional / Default is
"false"
        override_domain_name_servers => "true" | "false",                    #Optional / Default is
"false"
        override_options            => "true" | "false",                     #Optional / Default is
"false"
        override_preferred_lifetime => "true" | "false",                     #Optional / Default is
"false"
        override_valid_lifetime     => "true" | "false",                     #Optional / Default is
"false"
                                       $num                                  #Optional / Default is 27000
```

```
        preferred_lifetime          =>      ,
        valid_lifetime              => $num,                                    #Optional / Default is 43200
);
```

MODULE METHODS

The following functions can be applied to a DHCP IPv6 Fixed Address Template object.

Infoblox::DHCP::NetworkTemplate-> fixed_address_templates( )

Use this function to include a fixed address template in a network template on the Infoblox appliance. See Infoblox::DHCP::IPv6NetworkTemplate->fixed_address_templates() for parameters and return values.

Example

```
#Create the IPv6 network template that includes the IPv6 fixed address template
my $template = Infoblox::DHCP::Template->new(
  name                    => "custom_fa_template"
);
my $network_template = Infoblox::DHCP::IPv6NetworkTemplate->new(
  name                    => "custom_network_template",
  comment                 => "add network",
  fixed_address_templates => [$template]
);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that can be applied to a DHCP IPv6 fixed address template object.

Infoblox::Session->add( )

Use this method to add a DHCP IPv6 fixed address template object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a DHCP IPv6 Fixed Address Template object
my $fixed_addr_template = Infoblox::DHCP::IPv6FixedAddrTemplate->new(
);

#Submit for addition
my $response = $session->add($fixed_addr_template);
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 Fixed Address Template object:

name - Optional. The name of the fixed address template.
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
#Get DHCP IPv6 fixed address template object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6FixedAddrTemplate",
    name   => "custom_fa_template",
);

my @retrieved_objs = $session->get(
  object                => "Infoblox::DHCP::IPv6FixedAddrTemplate",
  extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->search( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP IPv6 Fixed Address Template object:

name    - Optional. The name of the fixed address template (regular expression).
comment - Optional. The comment of the fixed address template (regular expression).
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
#Search for a DHCP IPv6 fixed address template object through the session
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::IPv6FixedAddrTemplate",
    name    => "custom_fa_template",
    comment => ".*",
);

#Search for DHCP IPv6 Fixed Address Template objects that specify "Santa Clara" for the "Site"
extensible attribute
my @retrieved_objs = $session->search(
    object               => "Infoblox::DHCP::IPv6FixedAddrTemplate",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP IPv6 fixed address template object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Modify the pxe_lease_time
$fixed_addr_templateess->pxe_lease_time("1234");
#Submit modification
my $response = $session->modify( $fixed_addr_templateess );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP IPv6 fixed address template object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` to retrieve the DHCP IPv6 Fixed Address Template object, and then submit it for removal.

Example

```
#Get DHCP IPv6 fixed address template object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6FixedAddrTemplate",
    name   => "custom_fa_template",
);
#Find the desired object from the retrieved list.
my $desired_fixed_addr_template = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_fixed_addr_template );
```

METHODS

This section describes all the methods that can be used to set and retrieve the attribute values of a DHCP IPv6 Fixed Address Template object.

comment( )

Use this method to set or retrieve the descriptive comment of a DHCP IPv6 Fixed Address Template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
```

```
my $comment = $fixed_addr_template->comment();
#Modify comment
$fixed_addr_template->comment("Modified DHCP Fixed Address template object comment");
```

domain_name( )

Use this method to set or retrieve the domain_name value of a DHCP IPv6 Fixed Address Template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_domain_name attribute to "false".

Note that when domain_name is set to a defined value and override_domain_name is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name("testdomain"); $object->override_domain_name("false"); will set override_domain_name to "false", and the sequence $object->override_domain_name("false"); $object->domain_name("testdomain"); will result in override_domain_name="true".

Parameter

The client's domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name
my $domain_name = $fixed_addr_template->domain_name();
#Modify domain_name
$fixed_addr_template->domain_name("test_domain.com");
```

domain_name_servers( )

Use this method to set or retrieve the domain_name_servers value of a DHCP IPv6 fixed address template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name_servers method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_domain_name_servers attribute to "false".

Note that when domain_name_servers is set to a defined value and override_domain_name_servers is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name_servers("2001::1,2001::2"); $object->override_domain_name_servers("false"); will set override_domain_name_servers to "false", and the sequence $object->override_domain_name_servers("false"); $object->domain_name_servers("2001::1,2001::2"); will result in override_domain_name_servers="true".

Parameter

Comma-separated list of the IPv6 addresses of the DNS servers.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name_servers
my $domain_name_servers = $fixed_addr_template->domain_name_servers();
#Modify domain_name_servers
$fixed_addr_template->domain_name_servers("2001::1,2001::2");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP IPv6 Fixed Address Template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the

value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $fixed_addr_template->extensible_attributes();
#Modify extensible attributes
$fixed_addr_template->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [
'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the name of a DHCP IPv6 Fixed Address Template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the IPv6 fixed address template in string format and with a maximum of 64 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $fixed_address_template->name();
#Modify name
$fixed_address_template->name("custom_fa_template");
```

number_of_addresses( )

Use this method to set or retrieve the number of addresses in a DHCP IPv6 fixed address template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of addresses in this fixed address template.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get number_of_addresses
my $number_of_addresses = $fixed_address_template-> number_of_addresses();
#Modify number_of_addresses
$fixed_address_template->number_of_addresses("10");
```

offset( )

Use this method to set or retrieve the address offset of an IPv6 fixed address template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The address offset of this IPv6 fixed address template. The offset is calculated from the start address of the IPv6 network template that includes the fixed address template. For example, in the IPv6 network 2001:0:0:10::/8, if the offset is set to 10 and number_of_addresses is set to 10, then 10 fixed address objects will be created. The start address will be 2001:0:0:10::A and the end address will be 2001:0:0:10::14.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get offset
my $offset = $fixed_address_template->offset();
#Modify offset
$fixed_address_template->offset("10");
```

options( )

Use this method to set or retrieve the options of a DHCP IPv6 fixed address template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

See Infoblox::DHCP::Option for parameters and methods available for this object.

Setting this method to a defined value implicitly sets the override_options method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_options attribute to "false".

Note that when the options method is set to a defined value and override_options is set to "false", the last operation takes precedence. Thus the sequence $object->options([$opt]); $object->override_options("false"); will set override_options to "false", and the sequence $object->override_options("false"); $object->options([$opt]); will result in override_options="true".

Parameter

Options describe network configuration settings and various services available on the network. These options occur as variable-length fields at the end of DHCP messages.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $fixed_addr_template->options();
#Modify options
$fixed_addr_template->options([$option1, $option2]);
```

override_domain_name( )

The override_domain_name attribute controls whether the domain_name value in the DHCP IPv6 fixed address template is used, instead of the upper-level default.

The override_domain_name attribute can be specified explicitly. It is also set implicitly when domain_name is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for domain_name. Set the parameter to "false" to inherit the upper-level setting for domain_name.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_domain_name
my $override_domain_name=$fixed_addr_template->override_domain_name( );
#Modify override_domain_name
$fixed_addr_template->override_domain_name("true");
```

override_domain_name_servers( )

The override_domain_name_servers attribute controls whether the domain_name_servers value in the DHCP IPv6 fixed address template is used, instead of the upper-level default.

The override_domain_name_servers attribute can be specified explicitly. It is also set implicitly when domain_name_servers is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for domain_name_servers. Set the parameter to "false" to inherit the upper-level setting for domain_name_servers.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_domain_name_servers
my $override_domain_name_servers=$fixed_addr_template->override_domain_name_servers( );
#Modify override_domain_name_servers
$fixed_addr_template->override_domain_name_servers("true");
```

override_options( )

The override_options attribute controls whether the options in the DHCP IPv6 fixed address template are used, instead of the upper-level default.

The override_options attribute can be specified explicitly. It is also set implicitly when options are set to defined values.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for options. Set the parameter to "false" to inherit the upper-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_options
my $override_options=$fixed_addr_template->override_options( );
#Modify override_options
$fixed_addr_template->override_options("true");
```

override_preferred_lifetime( )

The override_preferred_lifetime attribute controls whether the preferred_lifetime value in the DHCP IPv6 fixed address template is used, instead of the upper-level default.

The override_preferred_lifetime attribute can be specified explicitly. It is also set implicitly when preferred_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for options. Set the parameter to "false" to inherit the upper-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_preferred_lifetime
my $override_preferred_lifetime=$fixed_addr_template->override_preferred_lifetime( );
#Modifying override_preferred_lifetime
```

```
$fixed_addr_template->override_preferred_lifetime("true");
```

override_valid_lifetime( )

The override_valid_lifetime attribute controls whether the valid_lifetime value in the DHCP IPv6 fixed address template object is used, instead of the upper-level default.

The override_valid_lifetime attribute can be specified explicitly. It is also set implicitly when valid_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for valid_lifetime. Set the parameter to "false" to inherit the upper-level setting for valid_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_valid_lifetime
my $override_valid_lifetime=$fixed_addr_template->override_valid_lifetime( );
#Modifying override_valid_lifetime
$fixed_addr_template->override_valid_lifetime("true");
```

preferred_lifetime( )

Use this method to set or retrieve the preferred lifetime of the DHCP IPv6 fixed address template.

Setting this method to a defined value implicitly sets the override_preferred_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_preferred_lifetime attribute to "false".

Note that when preferred_lifetime is set to a defined value and override_preferred_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->preferred_lifetime(20000); $object->override_preferred_lifetime("false"); will set override_preferred_lifetime to "false", and the sequence $object->override_preferred_lifetime("false"); $object->preferred_lifetime(20000); will result in override_preferred_lifetime="true".

Parameter

Preferred lifetime value in seconds. The default value is 27000. Note that the preferred lifetime cannot be greater than the valid lifetime.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get preferred_lifetime
my $preferred_lifetime = $fixed_addr_template->preferred_lifetime();
#Modify preferred_lifetime
$fixed_addr_template->preferred_lifetime(12345);
```

valid_lifetime( )

Use this method to set or retrieve the valid lifetime of the DHCP IPv6 fixed address template.

Setting this method to a defined value implicitly sets the override_valid_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_valid_lifetime attribute to "false".

Note that when valid_lifetime is set to a defined value and override_valid_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->valid_lifetime(20000); $object->override_valid_lifetime("false"); will set override_valid_lifetime to "false", and the sequence $object->override_valid_lifetime("false"); $object->valid_lifetime(20000); will result in override_valid_lifetime="true".

Parameter

Valid lifetime value in seconds. The default value is 43200.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get valid_lifetime
my $valid_lifetime = $fixed_addr_template->valid_lifetime();
#Modify valid_lifetime
$fixed_addr_template->valid_lifetime(123456);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, get, modify, and remove. Also, this sample includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);

unless ($session) {
    die("Construct session failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $option1 = Infoblox::DHCP::Option->new(
    "name"  => "dhcp6.fqdn",
    "value" => "mydomain.com"
);
unless($option1) {
    die("Construct option1 object failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option1 object created successfully\n";

my $option2 = Infoblox::DHCP::Option->new(
    "name"     => "dhcp6.info-refresh-time",
    "value"    => "1000",
);
unless($option2) {
    die("Construct option2 object failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option2 object created successfully\n";
```

#Create a DHCP IPv6 Fixed Address Template object

```
#Create a DHCP IPv6 Fixed Address Template object
my $fixed_addr_template = Infoblox::DHCP::IPv6FixedAddrTemplate->new(
    "name"                 => "custom_fa_template",
    "comment"              => "add IPv6 fixed address template",
    "domain_name"          => "test_domain.com",
    "number_of_addresses"   => 1,
    "offset"               => 10,
    "options"              => [ $option1, $option2 ],
    "preferred_lifetime"    => 1000,
    "valid_lifetime"        => 3600,
);
unless($fixed_addr_template) {
    die("Construct IPv6 Fixed Address template failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6 Fixed Address template object created successfully\n";

#Verify if the DHCP IPv6 Fixed Address template exists in the Infoblox appliance
my $object = $session->get(
    object => "Infoblox::DHCP::IPv6FixedAddrTemplate",
```

```
    name   => "custom_fa_template"
);

unless ($object) {
    print "IPv6 Fixed Address template does not exist on server, safe to add the IPv6 fixed address
template\n";
    $session->add($fixed_addr_template)
        or die("Add IPv6 Fixed Address template failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "DHCP IPv6 Fixed Address template added successfully\n";
```

#Search for a DHCP IPv6 Fixed Address Template object

```
#Search for a DHCP IPv6 fixed address template object through the session
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::IPv6FixedAddrTemplate",
    name   => "custom_fa_template",
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Search IPv6 Fixed Address template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search IPv6 Fixed Address template object found at least 1 matching entry\n";
```

#Get and modify a DHCP IPv6 Fixed Address Template object

```
#Get DHCP IPv6 fixed address template object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6FixedAddrTemplate",
    name   => "custom_fa_template",
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get IPv6 Fixed Address template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get IPv6 Fixed Address template object found at least 1 matching entry\n";

#Modify one of the attributes of the obtained IPv6 Fixed Address Template object
$object->valid_lifetime(1234);

#Apply the changes
$session->modify($object)
    or die("Modify IPv6 Fixed Address template failed: ",
    $session->status_code() . ":" . $session->status_detail());

print "DHCP IPv6 Fixed Address template added successfully\n";
```

#Create the IPv6 network template object and include the IPv6 fixed address template

```
my $template = Infoblox::DHCP::Template->new(
    name => "custom_fa_template",
);

my $network_template = Infoblox::DHCP::IPv6NetworkTemplate->new(
    name    => "custom_network_template",
    comment => "add network",
    fixed_address_templates => [$template]
);

unless($network_template) {
    die("Construct Network template object failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}

print "Network template object created successfully\n";

#Add the IPv6 network template object to the Infoblox appliance through a session
my $resp = $session->add($network_template)
    or die("Add Network object failed: ",
    $session->status_code() . ":" . $session->status_detail());

print "Network template object added to Infoblox appliance successfully\n";
```

#Remove an IPv6 fixed address template object

```
#Remove an IPv6 network template object
$session->remove($network_template)
    or die("Remove Network Template template object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Network Template object removed successfully \n";

#Get an IPv6 Fixed Address Template object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6FixedAddrTemplate",
    name   => "custom_fa_template",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get IPv6 Fixed Address template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get IPv6 Fixed Address template object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove IPv6 Fixed Address template object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "IPv6 Fixed Address template object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify(), Infoblox::DHCP::IPv6NetworkTemplate, Infoblox::DHCP::IPv6FixedAddr, Infoblox::DHCP::IPv6RangeTemplate,Infoblox::DHCP::Template,Infoblox::Session, Infoblox::DHCP::Option

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::IPv6Network - DHCP IPv6 Network object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - auto_create_reversezone( )
  - comment( )
  - ddns_domainname( )
  - ddns_enable_option_fqdn( )
  - ddns_generate_hostname( )
  - ddns_server_always_updates( )
  - ddns_ttl( )
  - disable( )
  - domain_name( )
  - domain_name_servers( )
  - enable_ddns( )
  - extensible_attributes( )
  - members( )
  - network( )
  - network_container( )
  - network_view( )
  - next_available_ip( )
  - options( )
  - override_ddns_domainname( )
  - override_ddns_enable_option_fqdn( )
  - override_ddns_generate_hostname( )
  - override_ddns_ttl( )
  - override_domain_name( )
  - override_domain_name_servers( )
  - override_enable_ddns( )
  - override_options( )
  - override_preferred_lifetime( )
  - override_recycle_leases( )
  - override_update_dns_on_lease_renewal( )
  - override_valid_lifetime( )
  - preferred_lifetime ( )
  - recycle_leases( )
  - shared_network_name( )
  - template( )
  - update_dns_on_lease_renewal( )
  - valid_lifetime ( )
  - zone_associations( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6Network - DHCP IPv6 Network object

DESCRIPTION

When DHCP services are configured on an appliance, a network that it serves must be defined. After a network is created, you can then create all the subnetworks individually or create a parent network that encompasses all the subnetworks. This object allows the user to create IPv6 networks. For IPv4 networks, see Infoblox::DHCP::Network object.

CONSTRUCTOR

```
  my $network = Infoblox::DHCP::IPv6Network->new(
                  network                          => $ipv6_subnet/prefix,              #Required,
when creating an IPv6 network
```

```
                auto_create_reversezone              => "true" | "false",                 #Optional /
Default is undefined
                comment                              => $string| undef,                   #Optional /
Default is undefined
                ddns_domainname                      => $string| undef,                   #Optional /
Default is undefined
                ddns_enable_option_fqdn              => "true" | "false" ,                #Optional /
Default is "false"
                ddns_generate_hostname               => "true" | "false" | undef,         #Optional /
Default is undefined
                ddns_server_always_updates           => "true" | "false",                 #Optional /
Default is "true"
                ddns_ttl                             => $num,                             #Optional /
Default is "0"
                disable                              => "true" | "false",                 #Optional /
Default is "false"
                domain_name                          => $string,                          #Optional /
Default is undefined
                domain_name_servers                  => "$ipv6addr,$ipv6addr,...",        #Optional /
Default is undefined
                enable_ddns                          => "true" | "false" | undef,         #Optional /
Default is undefined
                extensible_attributes                => { $string => $string | $num, $string => [
$string | $num, ... ], ... }, #Optional / Default is undefined.
                members                              => [ $Member1,$Member2,...],         #Optional /
Default is empty
                network_container                    => $ipv6_subnet/prefix,              #Optional /
Default is "/"
                network_view                         => $NetworkView,                     #Optional /
Default is "default" network view
                options                              => [$Option1,$Option2,...],          #Optional /
Default is empty list
                override_ddns_domainname             => "true" | "false",                 #Optional /
Default value is "false"
                override_ddns_enable_option_fqdn     => "true" | "false",                 #Optional /
Default value is "false"
                override_ddns_generate_hostname      => "true" | "false",                 #Optional /
Default value is "false"
                override_ddns_ttl                    => "true" | "false",                 #Optional /
Default value is "false"
                override_domain_name                 => "true" | "false",                 #Optional /
Default value is "false"
                override_domain_name_servers         => "true" | "false",                 #Optional /
Default value is "false"
                override_enable_ddns                 => "true" | "false",                 #Optional /
Default value is "false"
                override_options                     => "true" | "false",                 #Optional /
Default value is "false"
                override_preferred_lifetime          => "true" | "false",                 #Optional /
Default value is "false"
                override_recycle_leases              => "true" | "false",                 #Optional /
Default value is "false"
                override_update_dns_on_lease_renewal => "true" | "false",                 #Optional /
Default value is "false"
                override_valid_lifetime              => "true" | "false",                 #Optional /
Default value is "false"
                preferred_lifetime                   => $num,                             #Optional /
Default is "27000"
                recycle_leases                       => "true" | "false" | undef,         #Optional /
Default is undefined
                shared_network_name                  => $string| undef,                   #Optional /
Default is undefined
                template                             => $string| undef,                   #Optional /
Default is undefined
                update_dns_on_lease_renewal          => "true" | "false",                 #Optional /
Default value is "false"
                valid_lifetime                       => $num,                             #Optional /
Default is "43200"
                zone_associations                    => [$zone1, $zone2, ... ] | undef,   #Optional /
Default is an empty list
              );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that can apply to a DHCP IPv6 Network object.

Infoblox::Session->add( )

Use this method to add an IPv6 Network object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Adding an IPv6 network.
#Construct an object
my $network = Infoblox::DHCP::IPv6Network->new(
       network => "2001::/64",
);

# Submit for adding an IPv6 network
my $response = $session->add( $network );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching IPv6 Network objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 network object:

 comment               - Optional. A comment in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 network               - Optional. The IPv6 address and netmask of an IPv6 network in CIDR format.
 network_view          - Optional. The network view in which the network is located. By default, all
network views are searched.
```

Examples

```
 my  @result_array = $session->get(
      object        => "Infoblox::DHCP::IPv6Network" ,
      network       => "2001::/64",
      network_view => "default"
 );

 my  @result_array = $session->get(
      object                => "Infoblox::DHCP::IPv6Network" ,
      extensible_attributes => { 'Site' => 'Santa Clara' }
 );

 my  @result_array = $session->get(
      object  => "Infoblox::DHCP::IPv6Network" ,
      comment => "my-comment",
 );
```

Infoblox::Session->modify( )

Use this method to modify an IPv6 Network object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify comment.
$object->comment("This is a new comment");
# Submit modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my  @result_array = $session->get(

      object        => "Infoblox::DHCP::IPv6Network" ,
      network       => "2001::/64",
      network_view  => "default"
 );

# find the desired object from the retrieved list.
my $desired_network = $result_array[0];

# Submit for removal
my $response = $session->remove( $desired_network );
```

Infoblox::Session->search( )

Use this method to search for a DHCP IPv6 Network object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP IPv6 Network object:

 comment               - Optional. A comment in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 network               - Optional. The IPv6 address and netmask of an IPv6 network in CIDR format.
 network_view          - Optional. The network view in which the network is located. By default, all
network views are searched.
```

Examples

```
# search for all DHCP IPv6 Network objects
    my @retrieved_objs = $session->search(
      object       => "Infoblox::DHCP::IPv6Network",
      network      => "2001.*",
      network_view => "default"
    );

# search IPv6 network by comment
    my @retrieved_objs = $session->search(
      object    => "Infoblox::DHCP::IPv6Network",
      comment   => "my-comment",
    );

# search for all DHCP IPv6 Network objects defining "Santa Clara" for "Site" extensible attribute
    my @retrieved_objs = $session->search(
      object               => "Infoblox::DHCP::IPv6Network",
      extensible_attributes => { 'Site' => 'Santa Clara' }
    );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP IPv6 Network object

auto_create_reversezone( )

Use this method to set the auto_create_reversezone flag of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. This is a write-only attribute and cannot be retrieved after being set.

Parameter

Specify "true" to set the auto_create_reversezone flag or "false" to deactivate/unset it. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
# Modify auto_create_reversezone
$network_container->auto_create_reversezone("true");
```

comment( )

Use this method to set or retrieve a descriptive comment of an IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
```

```
my $comment = $network->comment();
# Modify comment
$network->comment("add an IPv6 network");
```

ddns_domainname( )

Use this method to set or retrieve the ddns_domainname value of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_ddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

The dynamic DNS domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_domainname
my $ddns_domainname = $network->ddns_domainname();
#Modify ddns_domainname
$network->ddns_domainname("test_domain.com");
```

ddns_enable_option_fqdn( )

Use this method to set or retrieve the ddns_enable_option_fqdn flag of a DHCP IPv6 Network object. This method controls whether the FQDN option sent by the client is to be used, or if the server can automatically generate the FQDN. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_enable_option_fqdn method to "true". Setting the parameter to undefined causes the appliance to use the upper-level setting and automatically resets the override_ddns_enable_option_fqdn attribute to "false".

Note that when ddns_enable_option_fqdn is set to a defined value and override_ddns_enable_option_fqdn is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_enable_option_fqdn("true"); $object->override_ddns_enable_option_fqdn("false"); will set override_ddns_enable_option_fqdn to "false", and the sequence $object->override_ddns_enable_option_fqdn("false"); $object->ddns_enable_option_fqdn("true"); will result in override_ddns_enable_option_fqdn="true".

Parameter

Specify "true" to generate hostname or "false" to deactivate/unset it.The default value for this field is undefined, which indicates that this attribute inherit the member level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_enable_option_fqdn
my $ddns_enable_option_fqdn = $network->ddns_enable_option_fqdn()
# Modify ddns_enable_option_fqdn
$network->ddns_enable_option_fqdn("true");
#Un-override
$network->ddns_enable_option_fqdn(undef);
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname flag of a DHCP IPv6 Network object. Enable this method to allow only the DHCP server for generating hostname. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_generate_hostname method to "true". Setting the parameter to undefined causes the appliance to use the upper-level setting and automatically resets the override_ddns_generate_hostname attribute to "false".

Note that when ddns_generate_hostname is set to a defined value and override_ddns_generate_hostname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_generate_hostname("true"); $object->override_ddns_generate_hostname("false"); will set override_ddns_generate_hostname to "false", and the sequence $object->override_ddns_generate_hostname("false"); $object->ddns_generate_hostname("true"); will result in override_ddns_generate_hostname="true".

Parameter

Specify "true" to generate a hostname or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute inherits the member-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_generate_hostname
my $ddns_generate_hostname = $network->ddns_generate_hostname()
# Modify ddns_generate_hostname
$network->ddns_generate_hostname("true");
#Un-override
$network->ddns_generate_hostname(undef);
```

ddns_server_always_updates( )

Use this method to set or retrieve the ddns_server_always_updates flag of a DHCP IPv6 Network object. Enable this method to allow only the DHCP server to update DNS, regardless of the requests from the DHCP clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the flag that enables the DHCP server to always updates DNS or "false" to deactivate/unset it.The default value for this field is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_server_always_updates
my $ddns_server_always_updates = $network->ddns_server_always_updates()
# Modify ddns_server_always_updates
$network->ddns_server_always_updates("true");
```

ddns_ttl( )

Use this method to set or retrieve the DNS update Time to Live (TTL) value of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_ttl method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_ttl attribute to "false".

Note that when ddns_ttl is set to a defined value and override_ddns_ttl is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_ttl(12); $object->override_ddns_ttl("false"); will set override_ddns_ttl to "false", and the sequence $object->override_ddns_ttl("false"); $object->ddns_ttl(12); will result in override_ddns_ttl="true".

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the update is cached. Zero indicates that the update is not cached. The default value is 0.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get DNS Update ttl
my $dns_ttl = $network->ddns_ttl();
#Modify DNS Update ttl
$network->ddns_ttl(1200);
```

disable( )

Use this method to set or retrieve the disable flag of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get disable
my $disable = $network->disable()
# Modify disable
$network->disable("true");
```

domain_name( )

Use this method to set or retrieve the domain_name value of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_domain_name attribute to "false".

Note that when domain_name is set to a defined value and override_domain_name is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name("testdomain"); $object->override_domain_name("false"); will set override_domain_name to "false", and the sequence $object->override_domain_name("false"); $object->domain_name("testdomain"); will result in override_domain_name="true".

Parameter

The name of the client's domain for resolving names in DNS in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name
my $domain_name = $network->domain_name();
#Modify domain_name
$network->domain_name("test_domain.com");
```

domain_name_servers( )

Use this method to set or retrieve the domain_name_servers value of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name_servers method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_domain_name_servers attribute to "false".

Note that when domain_name_servers is set to a defined value and override_domain_name_servers is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name_servers("2001::1,2001::2"); $object->override_domain_name_servers("false"); will set override_domain_name_servers to "false", and the sequence $object->override_domain_name_servers("false"); $object->domain_name_servers("2001::1,2001::2"); will result in override_domain_name_servers="true".

Parameter

The domain name server addresses in IPv6 address format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name_servers
my $domain_name_servers = $network->domain_name_servers();
#Modify domain_name_servers
$network->domain_name_servers("2001::1,2001::2");
```

enable_ddns( )

Use this method to set or retrieve the dynamic DNS updates flag of a DHCP IPv6 Network object. The DHCP server can send DDNS updates to DNS servers in the same grid and to external DNS servers. This setting overrides the member level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_enable_ddns method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_enable_ddns attribute to "false".

Note that when enable_ddns is set to a defined value and override_enable_ddns is set to "false", the last operation takes precedence. Thus the sequence $object->enable_ddns("true"); $object->override_enable_ddns("false"); will set override_enable_ddns to "false", and the sequence $object->override_enable_ddns("false"); $object->enable_ddns("true"); will result in override_enable_ddns="true".

Parameter

Specify "true" to set the dynamic DNS updates flag or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute inherit the member level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_ddns
my $enable_ddns = $network->enable_ddns()
# Modify enable_ddns
$network->enable_ddns("true");
# Un-override enable_ddns
$network->enable_ddns(undef);
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with an IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $network->extensible_attributes();
#Modify extensible attributes
$network->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

members( )

Use this method to set or retrieve the member that serves DHCP for an IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Member objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $member1 = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
);

#Create the Network object with this member
my $network = Infoblox::DHCP::Network->new(
    network => "2001::/64",
    members => [$member1],
);
```

network( )

Use this method to retrieve the network attribute of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the address and netmask of the network in CIDR format (example: 2001::/64).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network
my $network = $network->network();
```

network_container( )

Use this method to set or retrieve the network container of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the address and netmask of the network in CIDR format (example: 2001::/64).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network container
my $network_container = $network->network_container();
#Modify network container
$network->network_container("2001::/64");
```

network_view( )

Use this method to set or retrieve the network view of the IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the network is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $network->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$network->network_view($nview);
```

next_available_ip( )

Use this method to retrieve the next available IPv6 address in the network. Note that to use this method, the object MUST be retrieved through a get or search call, or it must have been created with an add call.

Parameter

This method does not accept input parameters.

Returns

This method returns the next available IPv6 address in the network.

Within a DHCP IPv6Network object, the next available IP address is the first address in the network that matches all of the following rules:

```
* There is no AAAA, PTR, fixed address or host address record with this IP address
* There is no bulk host record with this IP address
* There is no unmanaged record with this IP address
* The IP address is not the network or broadcast address of this network
* The IP address is not in any IPv6 DHCP range within this network (including any exclusion range)
```

Example

```
# Retrieve a particular network
my @retrieved_objs = $session->get(
                        object   => "Infoblox::DHCP::IPv6Network",
                        network  => "10:9::/64"
);

$network = $retrieved_objs[0];

# Print the next available IPv6 address in the network
print $network->next_available_ip() . "\n";
```

options( )

Use this method to set or retrieve the options value of a member DHCP server. DHCP options provide network configuration settings and various services available on the network for DHCP.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_options method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_options attribute to "false".

Note that when the options method is set to a defined value and override_options is set to "false", the last operation takes precedence. Thus the sequence $object->options([$opt]); $object->override_options("false"); will set override_options to "false", and the sequence $object->override_options("false"); $object->options([$opt]); will result in override_options="true".

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $network->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name => "dhcp6.info-refresh-time",
    value => 60
);

$network->options([$option1]);
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP IPv6 Network is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$network->override_ddns_domainname( );
#Modifying override_ddns_domainname
$network->override_ddns_domainname("true");
```

override_ddns_enable_option_fqdn( )

The override_ddns_enable_option_fqdn attribute controls whether the ddns_enable_option_fqdn value in the DHCP IPv6 Network is used, instead of the grid default.

The override_ddns_enable_option_fqdn attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_enable_option_fqdn. Set the parameter to "false" to inherit the grid-level setting for ddns_enable_option_fqdn.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_enable_option_fqdn
my $override_ddns_enable_option_fqdn=$network->override_ddns_enable_option_fqdn( );
#Modifying override_ddns_enable_option_fqdn
$network->override_ddns_enable_option_fqdn("true");
```

override_ddns_generate_hostname( )

The override_ddns_generate_hostname attribute controls whether the ddns_generate_hostname value in the DHCP IPv6 Network is used, instead of the grid default.

The override_ddns_generate_hostname attribute can be specified explicitly. It is also set implicitly when ddns_generate_hostname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_generate_hostname. Set the parameter to "false" to inherit the grid-level setting for ddns_generate_hostname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_generate_hostname
my $override_ddns_generate_hostname=$network->override_ddns_generate_hostname( );
#Modifying override_ddns_generate_hostname
$network->override_ddns_generate_hostname("true");
```

override_ddns_ttl( )

The override_ddns_ttl attribute controls whether the ddns_ttl value in the DHCP IPv6 Network is used, instead of the grid default.

The override_ddns_ttl attribute can be specified explicitly. It is also set implicitly when ddns_ttl is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_ttl. Set the parameter to "false" to inherit the grid-level setting for ddns_ttl.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_ttl
my $override_ddns_ttl=$network->override_ddns_ttl( );
#Modifying override_ddns_ttl
$network->override_ddns_ttl("true");
```

override_domain_name( )

The override_domain_name attribute controls whether the domain_name value in the DHCP IPv6 Network is used, instead of the grid default.

The override_domain_name attribute can be specified explicitly. It is also set implicitly when domain_name is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for domain_name. Set the parameter to "false" to inherit the grid-level setting for domain_name.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_domain_name
my $override_domain_name=$network->override_domain_name( );
```

```
#Modifying override_domain_name
$network->override_domain_name("true");
```

override_domain_name_servers( )

The override_domain_name_servers attribute controls whether the domain_name_servers value in the DHCP IPv6 Network is used, instead of the grid default.

The override_domain_name_servers attribute can be specified explicitly. It is also set implicitly when domain_name_servers is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for domain_name_servers. Set the parameter to "false" to inherit the grid-level setting for domain_name_servers.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_domain_name_servers
my $override_domain_name_servers=$network->override_domain_name_servers( );
#Modifying override_domain_name_servers
$network->override_domain_name_servers("true");
```

override_enable_ddns( )

The override_enable_ddns attribute controls whether the enable_ddns value in the DHCP IPv6 Network is used, instead of the grid default.

The override_enable_ddns attribute can be specified explicitly. It is also set implicitly when enable_ddns is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for enable_ddns. Set the parameter to "false" to inherit the grid-level setting for enable_ddns.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_enable_ddns
my $override_enable_ddns=$network->override_enable_ddns( );
#Modifying override_enable_ddns
$network->override_enable_ddns("true");
```

override_options( )

The override_options attribute controls whether the options value in the DHCP IPv6 Network is used, instead of the grid default.

The override_options attribute can be specified explicitly. It is also set implicitly when options is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_options
my $override_options=$network->override_options( );
#Modifying override_options
$network->override_options("true");
```

override_preferred_lifetime( )

The override_preferred_lifetime attribute controls whether the preferred_lifetime value in the DHCP IPv6 Network is used, instead of the grid default.

The override_preferred_lifetime attribute can be specified explicitly. It is also set implicitly when preferred_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_preferred_lifetime
my $override_preferred_lifetime=$network->override_preferred_lifetime( );
#Modifying override_preferred_lifetime
$network->override_preferred_lifetime("true");
```

override_recycle_leases( )

The override_recycle_leases attribute controls whether the recycle_leases value in the DHCP IPv6 Network is used, instead of the grid default.

The override_recycle_leases attribute can be specified explicitly. It is also set implicitly when recycle_leases is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_recycle_leases
my $override_recycle_leases=$network->override_recycle_leases( );
#Modifying override_recycle_leases
$network->override_recycle_leases("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the DHCP IPv6 Network is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$network->override_update_dns_on_lease_renewal( );
#Modifying override_update_dns_on_lease_renewal
$network->override_update_dns_on_lease_renewal("true");
```

override_valid_lifetime( )

The override_valid_lifetime attribute controls whether the valid_lifetime value in the DHCP IPv6 network is used, instead of the grid default.

The override_valid_lifetime attribute can be specified explicitly. It is also set implicitly when valid_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_valid_lifetime
my $override_valid_lifetime=$network->override_valid_lifetime( );
#Modifying override_valid_lifetime
$network->override_valid_lifetime("true");
```

preferred_lifetime ( )

Use this method to set or retrieve the preferred lifetime value of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_preferred_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_preferred_lifetime attribute to "false".

Note that when preferred_lifetime is set to a defined value and override_preferred_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->preferred_lifetime(691200); $object->override_preferred_lifetime("false"); will set override_preferred_lifetime to "false", and the sequence $object->override_preferred_lifetime("false"); $object->preferred_lifetime(691200); will result in override_preferred_lifetime="true".

Parameter

A 32-bit integer (valid range from 1 to 4294967295) that represents the lifetime, in seconds. The default value is 27000.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get Preferred lifetime
my $preferred_lifetime = $network->preferred_lifetime();
```

```
#Modify Preferred lifetime
$network->preferred_lifetime(864000);
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag at the network level.

If the flag is enabled, the leases are kept in the recycle bin until one week after expiration. When the flag is disabled, the leases are irrecoverably deleted.

Setting this method to a defined value implicitly sets the override_recycle_leases method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_recycle_leases attribute to "false".

Note that when recycle_leases is set to a defined value and override_recycle_leases is set to "false", the last operation takes precedence. Thus the sequence $object->recycle_leases("true"); $object->override_recycle_leases("false"); will set override_recycle_leases to "false", and the sequence $objct->override_recycle_leases("false"); $object->recycle_leases("true"); will result in override_recycle_leases="true".

Parameter

Specify "true" to enable the recycle_leases flag or "false" to disable it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $network->recycle_leases();
#Modify recycle_leases
$network->recycle_leases("false");
```

shared_network_name( )

Use this method to retrieve the name of the shared network associated with the DHCP IPv6 Network object.

Parameter

none

Returns

The method returns the name of shared network.

Example

```
#Get shared_network_name
my $shared_network_name = $network->shared_network_name();
```

template( )

Use this method to set or retrieve the network template name of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid name in sting format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get template
my $template = $network->template();
#Modify template
$network->template("Template_Name");
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$network->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$network->update_dns_on_lease_renewal("true");
```

valid_lifetime ( )

Use this method to set or retrieve the valid lifetime value of a DHCP IPv6 Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_valid_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_valid_lifetime attribute to "false".

Note that when valid_lifetime is set to a defined value and override_valid_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->valid_lifetime(691200); $object->override_valid_lifetime("false"); will set override_valid_lifetime to "false", and the sequence $object->override_valid_lifetime("false"); $object->valid_lifetime(691200); will result in override_valid_lifetime="true".

Parameter

A 32-bit integer (valid range from 1 to 4294967295) that represents the valid lifetime, in seconds. The default value is 43200.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get valid_lifetime
my $valid_lifetime = $network->valid_lifetime();
#Modify valid_lifetime
$network->valid_lifetime(864000);
```

zone_associations( )

Use this method to set or retrieve the associated zone information.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Zone object(s). The default value is an empty list. If the network has a parent, it is possible to use the zone associations of the parent by setting the parameter to undef.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Getting zone associations
my $zone_associations = $network->zone_associations();
my $zone1 = Infoblox::DNS::Zone->new(
      name     => "test1.com",
);
my $zone2 = Infoblox::DNS::Zone->new(
      name     => "test2.com",
   );
#Modifying zone_associations
$network->zone_associations([$zone1, $zone2]);
#Clear zone_assocations
$network->zone_associations([]);
##Remove the override for zone_associations
$network->zone_associations(undef)
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP IPv6 Network insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP IPv6 Network object

```
my  $network = Infoblox::DHCP::IPv6Network->new(
               network => "2001::/64",
               comment => "add network",
             );

my $response = $session->add($network)
       or die("Add Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network added successfully\n";

#Search for a specific Network

#Search the network
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::IPv6Network",
    network => "2001.*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search Network failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search Network found at least 1 matching entry\n";
```

#Get and modify a Network attribute

```
#Get IPv6 Network through the session
my @retrieved_objs = $session->get(
           object =>"Infoblox::DHCP::IPv6Network" ,
           network => "2001::"
        );

my $object = $retrieved_objs[0];

unless ($object) {
```

```
    die("Get Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network found at least 1 matching entry\n";

#Modify one of the attributes of the specified IPv6 Network

$object->comment("This is a new comment"); #Apply the changes

$session->modify($object)
    or die("Modify Network attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "Network object modified successfully \n";

#Resize an IPv6 Network

#Get IPv6 Network through the session
my @retrieved_objs = $session->get(
           object =>"Infoblox::DHCP::IPv6Network" ,
           network => "2001::/64"
        );
my $object = $retrieved_objs[0];

#Remove an IPv6 Network Object

 #Get IPv6 Network object through the session
 my @retrieved_objs = $session->get(
           object  => "Infoblox::DHCP::IPv6Network" ,
           network => "2001::"
       );

   my $object = $retrieved_objs[0];

   unless ($object) {
    die("Get Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
    }

print "Get Network object found at least 1 matching entry\n";

#Submit the object for removal

$session->remove($object)
    or die("Remove Network failed: ",
        $session->status_code() . ":" . $session->status_detail());

print "Network removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::View, Infoblox::DHCP::IPv6NetworkContainer, Infoblox::DHCP::Network, Infoblox::DHCP::NetworkContainer, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::IPv6NetworkContainer - DHCP IPv6 Network Container object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - auto_create_reversezone( )
  - comment( )
  - extensible_attributes( )
  - network( )
  - remove_subnets( )
  - network_container( )
  - network_view( )
  - zone_associations( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6NetworkContainer - DHCP IPv6 Network Container object

DESCRIPTION

A network can contain child networks. The network that contains the child network is called a network container. This object encapsulates an IPv6 network container object.

CONSTRUCTOR

```
  my $network = Infoblox::DHCP::IPv6NetworkContainer->new(
                  network                       => $ipv6_subnet/prefix,              #Required, when
creating a network container
                  auto_create_reversezone        => "true" | "false",                #Optional / Default
is "false"
                  comment                       => $string| undef,                   #Optional / Default
is undefined
                  remove_subnets                => "true" | "false",                 #Optional / Default
is undefined
                  network_container              => $ipv6_subnet/prefix,              #Optional / Default
is "/"
                  network_view                   => $NetworkView,                     #Optional / Default
is the system-defined default network view
                  zone_associations              => [$zone1, $zone2, ... ] | undef,    #Optional / Default
is an empty list
                  extensible_attributes          => { $string => $string | $num, $string => [ $string |
$num, ... ], ... } #Optional / Default is undefined
              );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a DHCP IPv6 Network Container object.

Infoblox::Session->add( )

Use this method to add a DHCP IPv6 Network Container object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
      #Adding an IPv6 network container.
      #Construct an object
      my $network_container = Infoblox::DHCP::IPv6NetworkContainer->new(
            network => "2001::/64",
```

```
);

# Submit for adding a network container
my $response = $session->add( $network_container );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching DHCP IPv6 Network Container objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 Network Container object:

 network               - Optional. The IPv6 address and netmask of a network in CIDR format.
 network_view          - Optional. The network view in which the network is located. By default, all
network views are searched.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 comment               - Optional. A comment in string format.
```

Examples

```
my  @result_array = $session->get(
     object        => "Infoblox::DHCP::IPv6NetworkContainer",
     network       => "2001::/64",
     network_view  => "default"
);

my  @result_array = $session->get(
     object                => "Infoblox::DHCP::IPv6NetworkContainer",
     extensible_attributes => { 'Site' => 'Santa Clara' }
);

my  @result_array = $session->get(
     object  => "Infoblox::DHCP::IPv6NetworkContainer",
     comment => "my-comment",
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP IPv6 Network Container object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify comment.
$network_container->comment("modified comment");
# Submit modification
my $response = $session->modify( $network_container );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my  @result_array = $session->get(

      object        => "Infoblox::DHCP::IPv6NetworkContainer" ,
      network       => "2001::/64",
      network_view  => "default"
 );

# Find the desired object in the retrieved list.
my $network_container = $result_array[0];

# Submit for removal
my $response = $session->remove( $network_container );
```

Infoblox::Session->search( )

Use this method to search for a DHCP IPv6 Network Container object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP IPv6 Network Container object:

  network               - Optional. The IPv6 address and netmask of a network in CIDR format (regular
expression).
  network_view          - Optional. The network view in which the network is located. By default, all
network views are searched.
  extensible_attributes - Optional. A hash reference containing extensible attributes.
  comment               - Optional. A comment in string format (regular expression).
```

Examples

```
# search for all DHCP IPv6 Network Container objects
    my @retrieved_objs = $session->search(
      object       => "Infoblox::DHCP::IPv6NetworkContainer",
      network      => "2001.*",
      network_view => "default"
    );

# search network by comment
    my @retrieved_objs = $session->search(
      object    => "Infoblox::DHCP::IPv6NetworkContainer",
      comment   => "my-comment",
    );

# search for all DHCP IPv6 Network Container objects defining "Santa Clara" for "Site" extensible
attribute
    my @retrieved_objs = $session->search(
      object                => "Infoblox::DHCP::IPv6NetworkContainer",
      extensible_attributes => { 'Site' => 'Santa Clara' }
    );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP IPv6 Network Container object.

auto_create_reversezone( )

Use this method to set the auto_create_reversezone flag of a DHCP IPv6 Network Container object.

Include the specified parameter to set the attribute value. This is a write-only attribute and cannot be retrieved after it is set.

Parameter

Specify "true" to set the auto_create_reversezone flag or "false" to deactivate/unset it. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
# Modify auto_create_reversezone
$network_container->auto_create_reversezone("true");
```

comment( )

Use this method to set or retrieve a descriptive comment for a DHCP IPv6 Network Container object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network container in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $network_container->comment();
# Modify comment
$network_container->comment("add a network");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP IPv6 Network Container object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $network_container->extensible_attributes();
#Modify extensible attributes
$network_container->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter',
'Tom' ] });
```

network( )

Use this method to retrieve the network attribute of a DHCP IPv6 Network Container object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the address and netmask of the network in CIDR format (example: 2001::/64).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network
my $network = $network_container->network();
```

remove_subnets( )

Use this method to set or retrieve the remove_subnets flag of a DHCP IPv6 Network Container object.

You can set this flag to false, if you want to remove the network, but keep all subnets under it. This flag is used only when removing DHCP IPv6 Network Container objects. It is not retrieved or used for any other operation.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "false" to keep the subnets, or "true" to remove them. The default value is undefined, which is to remove all subnets.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Set remove_subnets before submitting for removal
$network_container->remove_subnets("false");
#Remove the container of the subnets
$session->remove($network_container);
```

network_container( )

Use this method to set or retrieve the network container of a DHCP IPv6 Network Container object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the address and netmask of the network in CIDR format (example: 2001::/64).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network container
my $network_container = $network->network_container();
#Modify network container
$network->network_container("2001::/64");
```

network_view( )

Use this method to set or retrieve the network view of the DHCP IPv6 network container.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the system-defined default network view, which means the network is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $network_container->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$network_container->network_view($nview);
```

zone_associations( )

Use this method to set or retrieve the associated zone information.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Zone object(s). The default value is an empty list. If the network container has a parent, it is possible to use the zone associations of the parent by setting the parameter to undef.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Getting zone associations
my $zone_associations = $network_container->zone_associations();
my $zone1 = Infoblox::DNS::Zone->new(
        name     => "test1.com",
);
my $zone2 = Infoblox::DNS::Zone->new(
        name     => "test2.com",
    );
#Modifying zone_associations
$network_container->zone_associations([$zone1, $zone2]);
#Clear zone_assocations
$network_container->zone_associations([]);
##Remove the override for zone_associations
$network_container->zone_associations(undef)
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP IPv6 Network Container object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP IPv6 Network Container object

```
my $network_container = Infoblox::DHCP::IPv6NetworkContainer->new(
            network => "2001::/64",
            comment => "add network container",
             );

my $response = $session->add($network_container)
       or die("Add Network Container failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network Container added successfully\n";

#Search for a specific Network Container

#Search the network container
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::IPv6NetworkContainer",
    network => "2001.*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search Network Container failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search Network Container found at least 1 matching entry\n";
```

#Get and modify an IPv6 Network Container attribute

```
#Get Network Container through the session
my @retrieved_objs = $session->get(
           object =>"Infoblox::DHCP::IPv6NetworkContainer" ,
           network => "2001::/64"
        );

my $object = $retrieved_objs[0];

unless ($object) {
    die("Get Network Container object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network Container found at least 1 matching entry\n";
```

#Modify one of the attributes of the specified IPv6 network container attribute

```
$object->comment("modified comment"); #Apply the changes

$session->modify($object)
    or die("Modify Network Container attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "Network Container object modified successfully \n";
```

#Remove a Network Container Object

```
 #Get Network Container object through the session
 my @retrieved_objs = $session->get(
          object  => "Infoblox::DHCP::IPv6NetworkContainer" ,
          network => "2001::"
      );

   my $object = $retrieved_objs[0];

   unless ($object) {
    die("Get Network Container object failed: ",
        $session->status_code() . ":" . $session->status_detail());
    }

print "Get Network Container object found at least 1 matching entry\n";
```

#Submit the object for removal

```
$session->remove($object)
    or die("Remove Network Container failed: ",
        $session->status_code() . ":" . $session->status_detail());

print "Network Container removed successfully \n";
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::View, Infoblox::DHCP::IPv6Network, Infoblox::DHCP::Network, Infoblox::DHCP::NetworkContainer, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::IPv6NetworkTemplate - DHCP IPv6 Network Template object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
- METHODS
  - allow_any_netmask( )
  - auto_create_reversezone( )
  - cidr( )
  - comment( )
  - ddns_domainname( )
  - ddns_enable_option_fqdn( )
  - ddns_generate_hostname( )
  - ddns_server_always_updates( )
  - ddns_ttl( )
  - domain_name( )
  - domain_name_servers( )
  - enable_ddns( )
  - extensible_attributes( )
  - fixed_address_templates( )
  - ipv6prefix( )
  - members( )
  - name( )
  - options( )
  - override_ddns_domainname( )
  - override_ddns_enable_option_fqdn( )
  - override_ddns_generate_hostname( )
  - override_ddns_ttl( )
  - override_domain_name( )
  - override_domain_name_servers( )
  - override_enable_ddns( )
  - override_options( )
  - override_preferred_lifetime( )
  - override_recycle_leases( )
  - override_update_dns_on_lease_renewal( )
  - override_valid_lifetime( )
  - preferred_lifetime( )
  - range_templates( )
  - recycle_leases( )
  - update_dns_on_lease_renewal( )
  - valid_lifetime( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6NetworkTemplate - DHCP IPv6 Network Template object

DESCRIPTION

You can use a template to create IPv6 networks in a quick and consistent way. Networks created from a network template inherit all the properties defined in the network template, except for the comment and cidr if they are defined in the network.

CONSTRUCTOR

```
my $ipv6network_template = Infoblox::DHCP::IPv6NetworkTemplate->new(
        name                                => $string,                                    #Required
        allow_any_netmask                   => "true" | "false",
#Optional / Default is "false"
        auto_create_reversezone             => "true" | "false",
#Optional / Default is "false"
```

```
          cidr                                    => $num,                                        #Optional
/ Default is undefined
          comment                                 => $string,                                     #Optional
/ Default is empty
          ddns_domainname                         => $fqdn,                                       #Optional
/ Default is undefined
          ddns_enable_option_fqdn                 => "true" | "false" ,
#Optional / Default is "false"
          ddns_generate_hostname                  => "true" | "false" | undef,
#Optional / Default is undefined
          ddns_server_always_updates              => "true" | "false",
#Optional / Default is "true"
          ddns_ttl                                => $num,                                        #Optional
/ Default is 0
          domain_name                             => $fqdn,                                       #Optional
/ Default is undefined
          domain_name_servers                     => "$ipv6addr1,$ipv6addr2,...",
#Optional / Default is undefined
          enable_ddns                             => "true" | "false" | undef,
#Optional / Default is undefined
          extensible_attributes                   => { $string => $string | $num, $string => [ $string |
$num, ... ], ... },
                                                                                                  #Optional
/ Default is undefined.

          fixed_address_templates                 => [$DhcpTemplate1,$DhcpTemplate2,...],
#Optional / Default is empty
          ipv6prefix                              => $ipv6prefix,                                 #Optional
/ Default is undefined
          members                                 => [$Member1, $Member2, ...],
#Optional / Default is empty
          options                                 => [$Option1, $Option2, ...],
#Optional / Default is empty list
          override_ddns_domainname                => "true" | "false",
#Optional / Default is "false"
          override_ddns_enable_option_fqdn        => "true" | "false",
#Optional / Default is "false"
          override_ddns_generate_hostname         => "true" | "false",
#Optional / Default is "false"
          override_ddns_ttl                       => "true" | "false",
#Optional / Default is "false"
          override_domain_name                    => "true" | "false",
#Optional / Default is "false"
          override_domain_name_servers            => "true" | "false",
#Optional / Default is "false"
          override_enable_ddns                    => "true" | "false",
#Optional / Default is "false"
          override_options                        => "true" | "false",
#Optional / Default is "false"
          override_preferred_lifetime             => "true" | "false",
#Optional / Default is "false"
          override_recycle_leases                 => "true" | "false",
#Optional / Default is "false"
          override_update_dns_on_lease_renewal    => "true" | "false",
#Optional / Default is "false"
          override_valid_lifetime                 => "true" | "false",
#Optional / Default is "false"
          preferred_lifetime                      => $num,                                        #Optional
/ Default is 27000
          range_templates                         => [$RangeTemplate1,$RangeTemplate2,...],
#Optional / Default is empty
          recycle_leases                          => "true" | "false" | undef,
#Optional / Default is undefined
          update_dns_on_lease_renewal             => "true" | "false",
#Optional / Default is "false"
          valid_lifetime                          => $num,                                        #Optional
/ Default is 43200
           );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a DHCP IPv6 network template object.

Infoblox::Session->add( )

Use this method to add an IPv6 network template object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
          #Construct an object
```

```
my $ipv6network_template = Infoblox::DHCP::IPv6NetworkTemplate->new(
      name              => "custom_network_template",
      allow_any_netmask => "true",
);

# Submit for addition
my $response = $session->add( $ipv6network_template );
```

Infoblox::Session->get( )

Use this method to retrieve the matching IPv6 network template objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 network template object:

 name - Required. The name of network template.
```

Example

```
my  @result_array = $session->get(
     object  => "Infoblox::DHCP::IPv6NetworkTemplate" ,
     name    => "custom_network_template"
);
```

Infoblox::Session->search( )

Use this method to retrieve all the matching IPv6 network template objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP IPv6 network template object:

 name    - Optional. The name of the network template (regular
           expression).
 comment - Optional. The comment of the network template (regular
           expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my  @result_array = $session->search(
     object  => "Infoblox::DHCP::IPv6NetworkTemplate" ,
     name    => "custom_network_template"
 comment => ".*",
);

# search all Network Templates with a given extensible attribute 'Site'
my @retrieved_objs = $session->search(
  object => "Infoblox::DHCP::IPv6NetworkTemplate",
  extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an IPv6 network template in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Modify enable_ddns method
$object->enable_ddns("true");
# Submit modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve it, and then submit it for removal.

Example

```
# Get the object under the same name
my  @result_array = $session->get(

      object => "Infoblox::DHCP::IPv6NetworkTemplate" ,
```

```
        name   => "custom_network_template"
 );

# Find the desired object in the retrieved list
my $desired_network_template = $result_array[0];

# Submit for removal
my $response = $session->remove( $desired_network_template );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP IPv6 network template object.

allow_any_netmask( )

Use this method to set or retrieve the allow_any_netmask parameter of a DHCP IPv6 network template object. If you set this parameter to "true", then the template allows any netmask. A user must specify a netmask when creating a network using this template. If you set this parameter to false, you must specify the "netmask" property for the network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the allow_any_netmask parameter or specify "false" if you want to specify a particular netmask. The default value is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get allow_any_netmask
my $allow_any_netmask = $ipv6network_template->allow_any_netmask()
# Modify allow_any_netmask
$ipv6network_template->allow_any_netmask("true");
```

auto_create_reversezone( )

Use this method to set or retrieve the auto_create_reversezone flag of a DHCP IPv6 network template object. This flag indicates whether the appliance must automatically create the corresponding reverse zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to create a reverse zone automatically or "false" to disable the automatic creation of a reverse zone.The default value for this field is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get auto_create_reversezone
my $auto_create_reversezone = $ipv6network_template->auto_create_reversezone();
# Modify auto_create_reversezone
$ipv6network_template->auto_create_reversezone("true");
```

cidr( )

Use this method to set or retrieve the CIDR for the IPv6 network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the CIDR as a number from 1 to 128.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get CIDR
my $cidr = $ipv6network_template->cidr();
#Modify CIDR
$ipv6network_template->cidr("68");
```

comment( )

Use this method to set or retrieve a descriptive comment about a DHCP IPv6 network template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network template in string format and with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $ipv6network_template->comment();
# Modify comment
$ipv6network_template->comment("add a network");
```

ddns_domainname( )

Use this method to set or retrieve the ddns_domainname value of a DHCP IPv6 Network Tempate object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_ddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

The dynamic DNS domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_domainname
my $ddns_domainname = $ipv6network_template->ddns_domainname();
#Modify ddns_domainname
$ipv6network_template->ddns_domainname("test_domain.com");
```

ddns_enable_option_fqdn( )

Use this method to set or retrieve the ddns_enable_option_fqdn flag of a DHCP IPv6 network template object. This method controls whether the FQDN option sent by the client is to be used, or if the server can automatically generate the FQDN. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_enable_option_fqdn method to "true". Setting the parameter to undefined causes the appliance to use the upper-level setting and automatically resets the override_ddns_enable_option_fqdn attribute to "false".

Note that when ddns_enable_option_fqdn is set to a defined value and override_ddns_enable_option_fqdn is set to "false", the last

operation takes precedence. Thus the sequence $object->ddns_enable_option_fqdn("true"); $object->override_ddns_enable_option_fqdn("false"); will set override_ddns_enable_option_fqdn to "false", and the sequence $object->override_ddns_enable_option_fqdn("false"); $object->ddns_enable_option_fqdn("true"); will result in override_ddns_enable_option_fqdn="true".

Parameter

Specify "true" to generate the hostname or "false" to deactivate/unset it. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_enable_option_fqdn
my $ddns_enable_option_fqdn = $ipv6network_template->ddns_enable_option_fqdn()
# Modify ddns_enable_option_fqdn
$ipv6network_template->ddns_enable_option_fqdn("true");
#Remove override
$network_template->ddns_enable_option_fqdn(undef);
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname flag of a DHCP IPv6 network template object. Set this flag to allow only the DHCP server to generate a hostname. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_generate_hostname method to "true". Setting the parameter to undefined causes the appliance to use the upper-level setting and automatically resets the override_ddns_generate_hostname attribute to "false".

Note that when ddns_generate_hostname is set to a defined value and override_ddns_generate_hostname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_generate_hostname("true"); $object->override_ddns_generate_hostname("false"); will set override_ddns_generate_hostname to "false", and the sequence $object->override_ddns_generate_hostname("false"); $object->ddns_generate_hostname("true"); will result in override_ddns_generate_hostname="true".

Parameter

Specify "true" to generate the hostname or "false" to deactivate/unset it. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_generate_hostname
my $ddns_generate_hostname = $ipv6network_template->ddns_generate_hostname()
# Modify ddns_generate_hostname
$ipv6network_template->ddns_generate_hostname("true");
#Remove override
$network_template->ddns_generate_hostname(undef);
```

ddns_server_always_updates( )

Use this method to set or retrieve the ddns_server_always_updates flag of a DHCP IPv6 network template object. Set this flag to allow the DHCP server only to update DNS, regardless of the requests from DHCP clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set this flag and allow only the DHCP server to update DNS or "false" to deactivate/unset the flag. The default value is true.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_server_always_updates
my $ddns_server_always_updates = $ipv6network_template->ddns_server_always_updates()
# Modify ddns_server_always_updates
$ipv6network_template->ddns_server_always_updates("true");
```

ddns_ttl( )

Use this method to set or retrieve the DNS update Time to Live (TTL) value of a DHCP IPv6 network template object. The TTL value indicates the number of seconds the Infoblox appliance caches a DDNS update.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_ttl method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_ttl attribute to "false".

Note that when ddns_ttl is set to a defined value and override_dddns_ttl is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_ttl(12); $object->override_ddns_ttl("false"); will set override_ddns_ttl to "false", and the sequence $object->override_ddns_ttl("false"); $object->ddns_ttl(12); will result in override_ddns_ttl="true".

Parameter

A 32-bit integer, from 0 to 4294967295, that represents the duration, in seconds, that the update is cached. Zero indicates that the update will not be cached. The default value is 0.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get DNS Update TTL
my $dns_ttl = $ipv6network_template->ddns_ttl();
#Modify DNS Update TTL
$ipv6network_template->ddns_ttl(1200);
```

domain_name( )

Use this method to set or retrieve the domain_name value of a DHCP IPv6 network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_domain_name attribute to "false".

Note that when domain_name is set to a defined value and override_domain_name is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name("testdomain"); $object->override_domain_name("false"); will set override_domain_name to "false", and the sequence $object->override_domain_name("false"); $object->domain_name("testdomain"); will result in override_domain_name="true".

Parameter

The client's domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name
my $domain_name = $ipv6network_template->domain_name();
#Modify domain_name
$ipv6network_template->domain_name("test_domain.com");
```

domain_name_servers( )

Use this method to set or retrieve the domain_name_servers value of a DHCP IPv6 network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name_servers method to "true". Setting the parameter to

undefined causes the appliance to use the grid default and automatically resets the override_domain_name_servers attribute to "false".

Note that when domain_name_servers is set to a defined value and override_domain_name_servers is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name_servers("2001::1,2001::2"); $object->override_domain_name_servers("false"); will set override_domain_name_servers to "false", and the sequence $object->override_domain_name_servers("false"); $object->domain_name_servers("2001::1,2001::2"); will result in override_domain_name_servers="true".

Parameter

The domain name server addresses in IPv6 address format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name_servers
my $domain_name_servers = $ipv6network_template->domain_name_servers();
#Modify domain_name_servers
$ipv6network_template->domain_name_servers("2001::1,2001::2");
```

enable_ddns( )

Use this method to set or retrieve the dynamic DNS updates flag of a DHCP IPv6 network template object. The DHCP server can send DDNS updates to DNS servers in the same grid and to external DNS servers. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_enable_ddns method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_enable_ddns attribute to "false".

Note that when enable_ddns is set to a defined value and override_enable_ddns is set to "false", the last operation takes precedence. Thus the sequence $object->enable_ddns("true"); $object->override_enable_ddns("false"); will set override_enable_ddns to "false", and the sequence $object->override_enable_ddns("false"); $object->enable_ddns("true"); will result in override_enable_ddns="true".

Parameter

Specify "true" to set the dynamic DNS updates flag or "false" to deactivate/unset it. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_ddns
my $enable_ddns = $ipv6network_template->enable_ddns()
# Modify enable_ddns
$ipv6network_template->enable_ddns("true");
# Remove the enable_ddns override
$ipv6network_template->enable_ddns(undef);
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with an IPv6 network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $ipv6network_template->extensible_attributes();
#Modify extensible attributes
$ipv6network_template->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [
'Peter', 'Tom' ] });
```

fixed_address_templates( )

Use this method to set or retrieve the IPv6 fixed address template that is assigned to this IPv6 network template. When you create a network based on a network template that includes a fixed address template, the fixed addresses are created based on the associated fixed address template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference of Infoblox::DHCP::Template objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get fixed_address_templates
my $fixed_address_templates = $ipv6network_template->fixed_address_templates();
# Modify fixed_address_templates
my $tmpl = Infoblox::DHCP::Template->new(
   name => "custom_fixed_address_template"
);
$network_template->fixed_address_templates([$tmpl]);
```

ipv6prefix( )

Use this method to set or retrieve the IPv6 prefix of a DHCP IPv6 network template object. The prefix must belong to the list of global prefixes defined in the Infoblox::Grid::DHCP object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

IPv6 prefix from the list of global IPv6 prefixes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. Is.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6prefix
my $ipv6prefix = $ipv6network_template->ipv6prefix();
#Modify ipv6prefix
$griddhcp->ipv6prefix("2001::");
```

members( )

Use this method to set or retrieve the member assigned to the IPv6 network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Member objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $member1 = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
);

#Create a network template with this member
my $ipv6network_template = Infoblox::DHCP::IPv6NetworkTemplate->new(
    name    => "custom_network_template",
    members => [$member1],
);
```

name( )

Use this method to set or retrieve the name of the DHCP IPv6 network template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The network template name in string format and with a maximum of 64 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $ipv6network_template->name();
#Modify name
$ipv6network_template->name("custom_network_template");
```

options( )

Use this method to set or retrieve the options value of an IPv6 network template object. DHCP options provide network configuration settings and various services available on the network for DHCP. This setting overrides the upper-level setting. The default value is undefined which indicates that this attribute inherit the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_options method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_options attribute to "false".

Note that when the options method is set to a defined value and override_options is set to "false", the last operation takes precedence. Thus the sequence $object->options([$opt]); $object->override_options("false"); will set override_options to "false", and the sequence $object->override_options("false"); $object->options([$opt]); will result in override_options="true".

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $ipv6network_template->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name   => "dhcp6.rapid-commit",
    value  => "true"
);
$ipv6network_template->options([$option1]);
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP IPv6 Network Template is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$ipv6network_template->override_ddns_domainname( );
#Modifying override_ddns_domainname
$ipv6network_template->override_ddns_domainname("true");
```

override_ddns_enable_option_fqdn( )

The override_ddns_enable_option_fqdn attribute controls whether the ddns_enable_option_fqdn value in the DHCP IPv6 network template is used, instead of the grid default.

The override_ddns_enable_option_fqdn attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_enable_option_fqdn. Set the parameter to "false" to inherit the grid-level setting for ddns_enable_option_fqdn.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_ddns_enable_option_fqdn
my $override_ddns_enable_option_fqdn=$ipv6network_template->override_ddns_enable_option_fqdn( );
#Modify override_ddns_enable_option_fqdn
$ipv6network_template->override_ddns_enable_option_fqdn("true");
```

override_ddns_generate_hostname( )

The override_ddns_generate_hostname attribute controls whether the ddns_generate_hostname value in the DHCP IPv6 network template is used, instead of the grid default.

The override_ddns_generate_hostname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_generate_hostname. Set the parameter to "false" to inherit the grid-level setting for ddns_generate_hostname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_ddns_generate_hostname
my $override_ddns_generate_hostname=$ipv6network_template->override_ddns_generate_hostname( );
```

```
#Modify override_ddns_generate_hostname
$ipv6network_template->override_ddns_generate_hostname("true");
```

override_ddns_ttl( )

The override_ddns_ttl attribute controls whether the ddns_ttl value in the DHCP IPv6 network template object is used, instead of the grid default.

The override_ddns_ttl attribute can be specified explicitly. It is also set implicitly when ddns_ttl is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_ttl. Set the parameter to "false" to inherit the grid-level setting for ddns_ttl.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_ddns_ttl
my $override_ddns_ttl=$ipv6network_template->override_ddns_ttl( );
#Modify override_ddns_ttl
$ipv6network_template->override_ddns_ttl("true");
```

override_domain_name( )

The override_domain_name attribute controls whether the domain_name value in the DHCP IPv6 network template is used, instead of the grid default.

The override_domain_name attribute can be specified explicitly. It is also set implicitly when domain_name is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for domain_name. Set the parameter to "false" to inherit the grid-level setting for domain_name.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_domain_name
my $override_domain_name=$ipv6network_template->override_domain_name( );
#Modifying override_domain_name
$ipv6network_template->override_domain_name("true");
```

override_domain_name_servers( )

The override_domain_name_servers attribute controls whether the domain_name_servers value in the DHCP IPv6 network template is used, instead of the grid default.

The override_domain_name_servers attribute can be specified explicitly. It is also set implicitly when domain_name_servers is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for domain_name_servers. Set the parameter to "false" to inherit the grid-level setting for domain_name_servers.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_domain_name_servers
my $override_domain_name_servers=$ipv6network_template->override_domain_name_servers( );
#Modify override_domain_name_servers
$ipv6network_template->override_domain_name_servers("true");
```

override_enable_ddns( )

The override_enable_ddns attribute controls whether the enable_ddns value in the DHCP IPv6 network template is used, instead of the grid default.

The override_enable_ddns attribute can be specified explicitly. It is also set implicitly when enable_ddns is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for enable_ddns. Set the parameter to "false" to inherit the grid-level setting for enable_ddns.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_enable_ddns
my $override_enable_ddns=$ipv6network_template->override_enable_ddns( );
#Modify override_enable_ddns
$ipv6network_template->override_enable_ddns("true");
```

override_options( )

The override_options attribute controls whether the options in the DHCP IPv6 network template are used, instead of the grid default.

The override_options attribute can be specified explicitly. It is also set implicitly when options is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_options
my $override_options=$ipv6network_template->override_options( );
#Modify override_options
$ipv6network_template->override_options("true");
```

override_preferred_lifetime( )

The override_preferred_lifetime attribute controls whether the preferred_lifetime value in the DHCP IPv6 network template is used, instead of the grid default.

The override_preferred_lifetime attribute can be specified explicitly. It is also set implicitly when preferred_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_preferred_lifetime
my $override_preferred_lifetime=$ipv6network_template->override_preferred_lifetime( );
#Modify override_preferred_lifetime
$ipv6network_template->override_preferred_lifetime("true");
```

override_recycle_leases( )

The override_recycle_leases attribute controls whether the recycle_leases value in the DHCP IPv6 network template is used, instead of the grid default.

The override_recycle_leases attribute can be specified explicitly. It is also set implicitly when recycle_leases is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_recycle_leases
my $override_recycle_leases=$ipv6network_template->override_recycle_leases( );
#Modify override_recycle_leases
$ipv6network_template->override_recycle_leases("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the DHCP IPv6 network template is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$ipv6network_template-
>override_update_dns_on_lease_renewal( );
#Modify override_update_dns_on_lease_renewal
```

```
$ipv6network_template->override_update_dns_on_lease_renewal("true");
```

override_valid_lifetime( )

The override_valid_lifetime attribute controls whether the valid_lifetime value in the DHCP IPv6 network template object is used, instead of the upper-level default.

The override_valid_lifetime attribute can be specified explicitly. It is also set implicitly when valid_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for valid_lifetime. Set the parameter to "false" to inherit the upper-level setting for valid_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_valid_lifetime
my $override_valid_lifetime=$ipv6network_template->override_valid_lifetime( );
#Modify override_valid_lifetime
$ipv6network_template->override_valid_lifetime("true");
```

preferred_lifetime( )

Use this method to set or retrieve the valid lifetime of the DHCP IPv6 network template object.

Setting this method to a defined value implicitly sets the override_preferred_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_preferred_lifetime attribute to "false".

Note that when preferred_lifetime is set to a defined value and override_preferred_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->preferred_lifetime(20000); $object->override_preferred_lifetime("false"); will set override_preferred_lifetime to "false", and the sequence $object->override_preferred_lifetime("false"); $object->preferred_lifetime(20000); will result in override_preferred_lifetime="true".

Parameter

Preferred lifetime value in seconds. The default value is 27000. Note that the preferred lifetime cannot be greater than the valid lifetime.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get preferred_lifetime
my $preferred_lifetime = $ipv6network_template->preferred_lifetime();
#Modify preferred_lifetime
$ipv6network_template->preferred_lifetime(12345);
```

range_templates( )

Use this method to set or retrieve the IPv6 address range template assigned to this IPv6 network template object. When you create a network based on a network template object with a range template, the IPv6 address ranges are created based on the associated IPv6 address range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference of Infoblox::DHCP::Template objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get range_templates
my $range_templates = $ipv6network_template->range_templates();
#Modify range_templates
my $tmpl = Infoblox::DHCP::Template->new(
   name => "custom_range_template"
);
$ipv6network_template->range_templates([ $tmpl ]);
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag for an IPv6 network template object.

If the flag is enabled, the leases are kept in the recycle bin until one week after expiration. When the flag is disabled, the leases are irrecoverably deleted.

Setting this method to a defined value implicitly sets the override_recycle_leases method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_recycle_leases attribute to "false".

Note that when recycle_leases is set to a defined value and override_recycle_leases is set to "false", the last operation takes precedence. Thus the sequence $object->recycle_leases("true"); $object->override_recycle_leases("false"); will set override_recycle_leases to "false", and the sequence $objct->override_recycle_leases("false"); $object->recycle_leases("true"); will result in override_recycle_leases="true".

Parameter

Specify "true" to enable the recycle_leases flag or "false" to disable it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $ipv6network_template->recycle_leases();
#Modify recycle_leases
$ipv6network_template->recycle_leases("false");
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$ipv6network_template->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$ipv6network_template->update_dns_on_lease_renewal("true");
```

valid_lifetime( )

Use this method to set or retrieve the valid lifetime of the DHCP IPv6 network template object.

Setting this method to a defined value implicitly sets the override_valid_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_valid_lifetime attribute to "false".

Note that when valid_lifetime is set to a defined value and override_valid_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->valid_lifetime(20000); $object->override_valid_lifetime("false"); will set override_valid_lifetime to "false", and the sequence $object->override_valid_lifetime("false"); $object->valid_lifetime(20000); will result in override_valid_lifetime="true".

Parameter

Valid lifetime value in seconds. The default value is 43200.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get valid_lifetime
my $valid_lifetime = $ipv6network_template->valid_lifetime();
#Modify valid_lifetime
$ipv6network_template->valid_lifetime(12345);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

```
#Preparation prior to a DHCP IPv6 network template object insertion
```

#PROGRAM STARTS: Include all the modules that will be used use strict; use Infoblox; **#Create a session to the Infoblox DNSone appliance** my $session = Infoblox::Session->new(master => "192.168.1.2", username => "admin", password => "infoblox");

```
unless ($session) {
    die("Construct session failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $memberdhcp = Infoblox::DHCP::Member->new(
    name     => $host_name,
    ipv4addr => $node_ip
);
```

#Add IPv6 network template object

```
#Create a DHCP network template object

my $network_template = Infoblox::DHCP::IPv6NetworkTemplate->new(
    name                   => "custom_network_template",
    cidr                   => 64,
    comment                => "add IPv6 network template",
    members                => [ $memberdhcp ],
    enable_ddns            => "false",
    valid_lifetime         => 99999,
    auto_create_reversezone => "true",
);

my $response = $session->add($network_template)
    or die("Add Network template failed: ",
    $session->status_code() . ":" . $session->status_detail());

print "Network template added successfully\n";
```

#Get and modify an IPv6 network template attribute

```
#Get Network template through the session
my  @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6NetworkTemplate",
    name   => "custom_network_template"
);
```

```
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get Network template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network template found at least 1 matching entry\n";
```

#Search an IPv6 network template attribute

```
#Search Network template through the session
my  @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::IPv6NetworkTemplate",
    name   => "custom_network_template"
);

my $object = $retrieved_objs[0];

unless ($object) {
    die("Search Network template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search Network template found at least 1 matching entry\n";

#Modify attributes of the specified IPv6 network template

$object->enable_ddns("true"); #Apply the changes

$session->modify($object)
    or die("Modify Network template attribute failed: ",
    $session->status_code() . ":" . $session->status_detail());

print "Network template object modified successfully \n";
```

#Remove an IPv6 network template object

```
#Get IPv6 Network Template object through the session

my @retrieved_objs = $session->get(
    object  => "Infoblox::DHCP::IPv6NetworkTemplate",
    name    => "custom_network_template"
);

my $object = $retrieved_objs[0];

unless ($object) {
    die("Get IPv6 Network Template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Get IPv6 Network Template object found at least 1 matching entry\n";

#Submit the object for removal

$session->remove($object)
    or die("Remove IPv6 Network Template failed: ",
    $session->status_code() . ":" . $session->status_detail());

print "IPv6 Network Template removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::IPv6Network, Infoblox::Grid::DHCP, Infoblox::DHCP::Template, Infoblox::DHCP::Member, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::IPv6OptionDefinition - DHCP IPv6 option definition object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->search( )
  - Infoblox::Session->remove( )
- METHODS
  - name( )
  - code( )
  - space( )
  - type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6OptionDefinition - DHCP IPv6 option definition object

DESCRIPTION

An IPv6OptionDefinition defines an IPv6 DHCP option within a specific IPv6 option space. A custom option can be defined in the predefined DHCP option space or in the user-defined vendor option space. To define an option, add the option definition to the required option space.

CONSTRUCTOR

```
my $ipv6od = Infoblox::DHCP::IPv6OptionDefinition->new(
                name  => $string, #Required
                code  => $num,    #Optional / Default is 1
                space => $string, #Optional / Default is DHCP
                type  => $string, #Optional / Default is string
            );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that can be applied to a DHCP IPv6 option definition object.

Infoblox::Session->add( )

Use this method to add an IPv6 option definition object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Create a DHCP IPv6OptionDefinition object.
my $ipv6od = Infoblox::DHCP::IPv6OptionDefinition->new(
                space => "SUNW",
                name  => "JumpStart-server",
                code  => "6",
                type  => "text"
            );
# Submit for adding
my $response = $session->add( $ipv6od );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching DHCP IPv6 option definition objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 option definition object:
```

```
  name  - A name for the option definition in string format (regular expression).
  code  - Option number. Either a name or code is required.
  space - Optional. The option space in which the option is defined. The default is the "DHCPv6"
option space.
```

Example

```
#Getting option by name
my  @result_array = $session->get(
                    object   => "Infoblox::DHCP::IPv6OptionDefinition",
                    space    => "SUNW",
                    name     => "JumpStart-server",
                    );

#Getting option by code
my @result_array = $session->get(
                    object   => "Infoblox::DHCP::IPv6OptionDefinition",
                    space    => "DHCPv6",
                    code     => 176,
                    );
```

Infoblox::Session->modify( )

Use this method to modify a DHCP IPv6 option definition object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify code
$object->code(20);
# Submit modification
my $response = $session->modify($object);
```

Infoblox::Session->search( )

Use this method to retrieve all the matching DHCP IPv6 option definition objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 option definition object:

  name  - A name for the option definition in string format (regular expression).
  code  - Option number. Either a name or code is required.
  space - Optional. The option space in which the option is defined. The default is the "DHCPv6"
option space.
```

Example

```
#Getting option by name
my  @result_array = $session->search(
                    object   => "Infoblox::DHCP::IPv6OptionDefinition",
                    space    => "SUNW",
                    name     => "JumpStart-server",
                    );

#Getting option by code
my @result_array = $session->search(
                    object   => "Infoblox::DHCP::IPv6OptionDefinition",
                    space    => "DHCPv6",
                    code     => 176,
                    );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP IPv6 option definition object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` to retrieve the specific object, and then submit this object for removal.

Example

```
my  @result_array = $session->get(
                    object   => "Infoblox::DHCP::IPv6OptionDefinition",
                    space    => "SUNW",
                    name     => "JumpStart-server",
```

```
                        );

# find the desired object from retrieved list.
my $desired_obj = $result_array[0];

# Submit for removal
my $response = $session->remove($desired_obj);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP IPv6 option definition object.

name( )

Use this method to set or retrieve the name of a DHCP IPv6 option definition object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $od_name = $ipv6od->name();
# Modify name
$ipv6od->name("Start-server");
```

code( )

Use this method to set or retrieve the code of a DHCP IPv6 option definition object. An option code number is used to identify the DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number for the option code. Value must be between 1 and 254. Values 2, 7, 12, 14, 21-14, 27-34, and 39 are predefined. These option codes have corresponding predefined Types. Therefore, you cannot change the predefined Types.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get code
my $od_code = $ipv6od->code();
# Modify code
$ipv6od->code(14);
```

space( )

Use this method to set or retrieve the space of a DHCP IPv6 option definition object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the space name in string format.

Returns

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get space
```

```
my $od_space = $ipv6od->space();
# Modify space
$ipv6od->space("SUNW");
```

type( )

Use this method to set or retrieve the data type of the grid DHCP IPv6 option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The accepted types are the following:

```
  "boolean"                 - For "true" or "false" boolean values.

  "ip-address"              - For a single IPv6 address. An IPv6 address is a 128-bit number.
                              It consist of eight groups of hexadecimal digits separated
                              by colon (example: "2001:1dfc:2dfb:0:c1:0:a0:a2"). A group of all zeros
                              can be replaced by a double colon (e.g. 2001:1dfc:2dfb::1 is equivalent
                              to 2001:1dfc:2dfb:0:0:0:0:1).

  "array of ip-address"     - For a list of IPv6 addresses, each separated by a comma.
                              (example: "2002:d0:d2::1, 2002:d0:d2::3, 2001::3")

  "string"                  - For value in ASCII text string or
                              a list of hexadecimal characters (0-9, a-f, or A-F) without spaces and
separated by colons.

  "text"                    - For value in ASCII text string.

  "domain-name"             - For domain name in ASCII text format

  "domain-list"             - For a list of domain names, each separated by a comma.

  "8-bit unsigned integer"  - For numeric value between 0 and 255.

  "16-bit unsigned integer" - For numeric value between 0 and 65535.

  "32-bit unsigned integer" - For numeric value between 0 and 4294967295.

  "8-bit signed integer"    - For numeric value between -128 and 127.

  "16-bit signed integer"   - For numeric value between -32768 and 32767.

  "32-bit signed integer"   - For numeric value between -2147483648 and 2147483647.

  "array of 8-bit unsigned integer"  - For a list of numeric values between 0 and 255, each
separated by a comma.

  "array of 16-bit unsigned integer" - For a list of numeric values between 0 and 65535, each
separated by a comma.

  "array of 32-bit unsigned integer" - For a list of numeric values between 0 and 4294967295, each
separated by a comma.

  "array of 8-bit integer"           - For a list of numeric values between -128 and 127, each
separated by a comma.

  "array of 16-bit integer"          - For a list of numeric values between -32768 and 32767, each
separated by a comma.

  "array of 32-bit integer"          - For a list of numeric values between -2147483648 and
2147483647, each separated by a comma.
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get type
my $od_type = $ipv6od->type();
# Modify type
$ipv6od->type("string");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP Network insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $ipv6od = Infoblox::DHCP::IPv6OptionSpace->new("name" => "SUNW");
my $response = $session->add($ipv6od);
print "Add option space operation success \n" if ($response);
print "Add option space operation failed  \n" if (not $response);
```

#Create a DHCP IPv6 OptionDefinition object.

```
my $ipv6od = Infoblox::DHCP::OptionDefinition->new(
                                 space => "SUNW",
                                 name  => "JumpStart-server",
                                 code  => "6",
                                 type  => "text"
                              );
unless ($ipv6od) {
    die("Construct DHCP IPv6OptionDefinition failed: ",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6OptionDefinition object created successfully\n";

#Add the DHCP IPv6OptionDefinition object to Infoblox appliance through a session
my $response = $session->add( $ipv6od )
   or die("Add a DHCP IPv6OptionDefinition object failed: ",
          $session->status_code() . ":" . $session->status_detail());

print "Add a DHCP IPv6OptionDefinition object successfully\n";
```

#Get and modify a DHCP OptionDefinition object attribute

```
#Get Network through the session
my @retrieved_objs = $session->get(
               object   => "Infoblox::DHCP::IPv6OptionDefinition",
               space  => "SUNW",
               name   => "JumpStart-server",
            );

my $object = $retrieved_objs[0];

unless ($object) {
    die("Get DHCP IPv6OptionDefinition object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP IPv6OptionDefinition object found at least 1 matching entry\n";

#Modify one of the attributes of the specified Network attribute

$object->code(20);    #Apply the changes

$session->modify($object)
    or die("Modify DHCP IPv6OptionDefinition attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "DHCP IPv6OptionDefinition object modified successfully \n";
```

#Remove a DHCP IPv6OptionDefinition Object

```
#Get DHCP IPv6OptionDefinition object through the session
my @retrieved_objs = $session->get(
               object =>"Infoblox::DHCP::IPv6OptionDefinition",
               space  => "SUNW",
               name   => "JumpStart-server",
```

```
            );

 my $object = $retrieved_objs[0];

 unless ($object) {
     die("Get DHCP IPv6OptionDefinition object failed: ",
         $session->status_code() . ":" . $session->status_detail());
 }

 print "Get DHCP IPv6OptionDefinition object found at least 1 matching entry\n";

#Submit the object for removal

 $session->remove($object)
     or die("Remove Network failed: ",
         $session->status_code() . ":" . $session->status_detail());

 print "DHCP IPv6OptionDefinition removed successfully \n";

 #Remove option space.

 my $response = $session->remove( object => "Infoblox::DHCP::IPv6OptionSpace" ,
                                  name   => "SUNW",
                                );
 print "Remove option space success \n" if ($response );
 print "Remove option space failed  \n" if ( not $response );

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::IPv6OptionSpace, Infoblox::DHCP::IPv6Option, Infoblox::Session->get(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 2010 Infoblox Inc.

Infoblox::DHCP::IPv6OptionSpace - DHCP IPv6 OptionSpace object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - comment( )
  - enterprise_number( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6OptionSpace - DHCP IPv6 OptionSpace object.

DESCRIPTION

An Option Space defines a namespace in which vendor options can be defined. This object is for an IPv6 option space. To define a specific vendor option space, add an option space to DHCP.

CONSTRUCTOR

```
 my $ipv6os = Infoblox::DHCP::IPv6OptionSpace->new(
       name               => $string, #Required
       enterprise_number  => $num,    #Required, if this is a vendor option space.  It is optional
otherwise.
       comment            => $string, #Optional, default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that can be applied to a DHCP IPv6 OptionSpace object.

Infoblox::Session->add( )

Use this method to add a DHCP IPv6 OptionSpace object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
       #Construct a DHCP IPV6 OptionSpace object
       my $ipv6os = Infoblox::DHCP::IPv6OptionSpace->new(
             name => "infospace",
             comment => "some comment",
             enterprise_number => 123,
       );
       #Submit for addition
       my $response = $session->add($ipv6os)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
       Apply the following attributes to get a specific DHCP IPv6 OptionSpace object:

        name     - Required. Option space name in string format.
        comment  - Optional. Option space comment.
        enterprise_number - Optional. Enterprise number.
```

Example

```
#Get DHCP IPv6 OptionSpace object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6OptionSpace",
    name   => "infospace",
    enterprise_number => 123,
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP IPv6 OptionSpace object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the name.
$ipv6os->name("info_space");
#Submit modification
my $response = $session->modify( $ipv6os );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP IPv6 OptionSpace object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` to retrieve the specific DHCP IPv6 OptionSpace object, and then submit this object for removal.

Example

```
#Get DHCP IPv6 OptionSpace object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6OptionSpace",
    name   => "info_space",
    enterprise_number => 123,
);
#Find the desired object from the retrieved list.
my $desired_option_space = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_option_space );
```

Infoblox::Session->search( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 OptionSpace object:

 name     - Optional. Option space name in string format.
 comment  - Optional. Option space comment (regular expression).
 enterprise_number - Optional. Enterprise number.
```

Example

```
#Get DHCP IPv6 OptionSpace object through the session
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::IPv6OptionSpace",
    comment => ".*comm.*"
);
```

METHODS

This section describes all the methods that can used to set and retrieve the attribute values of a DHCP IPv6 OptionSpace object.

name( )

Use this method to set or retrieve name of IPv6 option space object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name in string format with a maximum of 256 bytes.

Returns

If parameter is specified, the method returns true when the modification succeeds, and returns false when the operation fails.

If parameter is not specified, the method returns the attribute value.

Example

```
#Get name
my $name = $ipv6os->name();
#Modify name
$ipv6os->name("info_space");
```

comment( )

Use this method to set or retrieve a comment about the IPv6 option space object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If the parameter is specified, the method returns true when the modification succeeds, and returns false when the operation fails.

If the parameter is not specified, the method returns the attribute value.

Example

```
#Get comment
my $comment = $ipv6os->comment();
#Modify comment
$ipv6os->comment("my comment");
```

enterprise_number( )

Use this method to set or retrieve the enterprise number for the IPv6 option space object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer that represents the enterprise number.

Returns

If the parameter is specified, the method returns true when the modification succeeds, and returns false when the operation fails.

If the parameter is not specified, the method returns the attribute value.

Example

```
#Get enterprise_number
my $enterprise_number = $ipv6os->enterprise_number();
#Modify enterprise_number
$ipv6os->enterprise_number(345);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an IPv6OptionSpace object, such as get, add, modify and remove. Also sample includes error handling for the operations.

#Preparation prior to a DHCP IPv6OptionSpace object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);
```

```
unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create an IPv6OptionSpace object

```
my $ipv6os = Infoblox::DHCP::IPv6OptionSpace->new (
       "name" => "infospace"
);

unless($ipv6os) {
     die("Construct IPv6 option_space filter failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "IPv6OptionSpace object created successfully\n";

#Verify if the DHCP IPv6OptionSpace exists in the Infoblox appliance
my $object = $session->get(
       object => "Infoblox::DHCP::IPv6OptionSpace",
       name   => "infospace"
       enterprise_number => 123,
);
unless ($object) {
   print "IPv6 option space does not exist on server, safe to add the option_space\n";
   $session->add($ipv6os)
      or die("Add IPv6 option space failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "DHCP IPv6 OptionSpace added successfully\n";
```

#Get and Modify an IPv6OptionSpace object

```
#Get IPv6OptionSpace object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6OptionSpace",
    name   => "infospace"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get IPv6OptionSpace object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get IPv6OptionSpace object found at least 1 matching entry\n";

#Modify name of the obtained IPv6OptionSpace object
$object->name("info_space");

#Apply the changes
$session->modify($object)
    or die("Modify IPv6OptionSpace object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "IPv6OptionSpace object modified successfully \n";
```

#Remove an IPv6OptionSpace object

```
#Get IPv6OptionSpace object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6OptionSpace",
    name   => "info_space"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get IPv6OptionSpace object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get IPv6OptionSpace object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove IPv6OptionSpace object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "IPv6OptionSpace object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::IPv6OptionDefinition, Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session

---

COPYRIGHT

Copyright (c) 2010 Infoblox Inc.

Infoblox::DHCP::IPv6Range - DHCP IPv6 Range object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - address_type( )
  - comment( )
  - disable( )
  - end_addr( )
  - exclude( )
  - extensible_attributes( )
  - ipv6_end_prefix( )
  - ipv6_prefix_bits( )
  - ipv6_start_prefix( )
  - member( )
  - name( )
  - network( )
  - network_view( )
  - next_available_ip( )
  - override_recycle_leases( )
  - recycle_leases( )
  - server_association_type( )
  - start_addr( )
  - template( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6Range - DHCP IPv6 Range object.

DESCRIPTION

A DHCP IPv6 range defines the specified range of IP addresses in an IPv6 network. A DHCP IPv6 range should be added for an IPv6 network so the Infoblox appliance can assign IP addresses within that specified range to DHCP clients. If the client is on an IPv6 network that is assigned a DHCP IPv6 range, the device distributes an available IP address from that range to the DHCP client, or to a DHCP relay agent if the request came through an agent. The DHCP IPv6 range should also be assigned with a device. If devices are in a grid, the particular member serving DHCP for the DHCP IPv6 range must be specified. If the server is an independent device, this device must be specified as the member that serves the DHCP IPv6 range.

CONSTRUCTOR

```
 my $dhcp_ipv6_range = Infoblox::DHCP::IPv6Range->new(
     network                         => $network,                                  #Required
     ipv6_end_prefix                 => $ipv6addr,                                 #Optional / See NOTE
below
     ipv6_prefix_bits                => $num,                                      #Optional / See NOTE
below
     ipv6_start_prefix               => $ipv6addr,                                 #Optional / See NOTE
below
     end_addr                        => $ipv6addr,                                 #Optional / See NOTE
below
     start_addr                      => $ipv6addr,                                 #Optional / See NOTE
below
     address_type                    => "ADDRESS" | "PREFIX" | "BOTH",             #Optional / Default is
undefined
     comment                         => $string,                                   #Optional / Default is
undefined
     disable                         => "true" | "false",                          #Optional / Default is
"false"
     exclude                         => [$ExclusionRange1,$ExclusionRange2,...],   #Optional / Default is
empty
```

```
    extensible_attributes          => { $string => $string | $num, $string => [ $string | $num, ... ],
... } #Optional / Default is undefined
    member                         => $Member,                                  #Optional / Default is
undefined
    name                           => $string,                                  #Optional / Default is
undefined
    network_view                   => $NetworkView,                             #Optional / Default is
system-defined default network view
    override_recycle_leases        => "true" | "false"                          #Optional / Default is
"false"
    recycle_leases                 => "true" | "false",                         #Optional / Default is
"true"
    server_association_type        => "NONE" | "MEMBER"                         #Optional / Default is
undefined
    template                       => $string,                                  #Optional / Default is
undefined
 );
```

NOTE: one or both of the start_addr/end_addr and ipv6_start_prefix/ipv6_end_prefix/ipv6_prefix_bits parameter sets is required.

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a DHCP IPv6 range object.

Infoblox::Session->add( )

Use this method to add a DHCP IPv6 range object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $dhcp_ipv6range = Infoblox::DHCP::IPv6Range->new(
                                                 start_addr => "2001:db8:0:c000:0001::",
                                                 end_addr   => "2001:db8:0:c000:0021::",
                                                 network    => "2001:db8:0:c000::/51",
);

# Submit for addition
my $response = $session->add( $dhcp_ipv6range );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 range object:

  start_addr             - Optional. Starting IP address of the DHCP IPv6 range within the network.
  end_addr               - Optional. Ending IP address of the DHCP IPv6 range within the network.
  ipv6_start_prefix      - Optional. Starting prefix of the DHCP IPv6 range.
  ipv6_end_prefix        - Optional. Ending prefix of the DHCP IPv6.
  network_view           - Optional. Network view in which the DHCP IPv6 range is located. By
default, all network views are searched.
  extensible_attributes  - Optional. A hash reference containing extensible attributes.
  member                 - Optional. The member serving the range. The valid value is an
Infoblox::DHCP::Member object.
  name                   - Optional. The name of the DHCP IPv6 Range.
  network                - Optional. The IPv6 Network the range resides in.
  comment                - Optional. The descriptive comment.
```

Examples

```
my @retrieved_objs = $session->get(
                     object     => "Infoblox::DHCP::IPv6Range",
                     start_addr => "2001:db8:0:c000:0001::" );

my @retrieved_objs = $session->get(
                     object       => "Infoblox::DHCP::IPv6Range",
                     start_addr   => "2001:db8:0:c000:0001::",
                     network_view => "default" );

my @retrieved_objs = $session->get(
                     object                => "Infoblox::DHCP::IPv6Range",
                     extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a DHCP IPv6 range object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$dhcp_ipv6range->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $dhcp_ipv6range );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP IPv6 range object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP IPv6 range object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::IPv6Range",
    start_addr   => "2001:db8:0:c000:0001::",
    newtork_view => "default" );
# Find the desired object from the retrieved list.
my $desired_dhcp_range = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_dhcp_range );
```

Infoblox::Session->search( )

Use this method to search for DHCP IPv6 range objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP IPv6 range object:

  start_addr             - Optional. Starting IP address of the DHCP IPv6 range within the network
(regular expression).
  end_addr               - Optional. Ending IP address of the DHCP IPv6 range within the network
(regular expression).
  ipv6_start_prefix      - Optional. Starting prefix of the DHCP IPv6 range (regular expression).
  ipv6_end_prefix        - Optional. Ending prefix of the DHCP IPv6 (regular expression).
  network_view           - Optional. Network view in which the DHCP IPv6 range is located. By
default, all network views are searched.
  extensible_attributes  - Optional. A hash reference containing extensible attributes.
  member                 - Optional. The member serving the range. The valid value is an
Infoblox::DHCP::Member object.
  name                   - Optional. The name of the DHCP IPv6 Range (regular expression).
  network                - Optional. The IPv6 Network the range resides in (regular expression).
  comment                - Optional. The descriptive comment (regular expression).
```

Examples

```
# search for DHCP IPv6 range object with start address having last octet as 1
my @retrieved_objs = $session->search(
    object     => "Infoblox::DHCP::IPv6Range",
    network       => "2001:db8:0:c000::/51",
    network_view  => "default",
    start_addr    => "0021",
    comment       => ".*comment" );

# search for DHCP IPv6 range objects defining "Santa Clara" for "Site" extensible attribute
my @retrieved_objs = $session->search(
    object                => "Infoblox::DHCP::IPv6Range",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DHCP IPv6 range.

address_type( )

Use this method to set or retrieve the address type of a DHCP IPv6 Range object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "ADDRESS", "PREFIX", or "BOTH". When the address type is "ADDRESS", values for the 'start_addr' and 'end_addr' members are required. When the address type is "PREFIX", values for 'ipv6_start_prefix', 'ipv6_end_prefix', and 'ipv6_prefix_bits' are required. When the address type is "BOTH", values for 'start_addr', 'end_addr', 'ipv6_start_prefix', 'ipv6_end_prefix', and 'ipv6_prefix_bits' are all required.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get address_type
my $address_type = $dhcp_ipv6range->address_type();
#Modify address_type
$dhcp_ipv6range->address_type("PREFIX");
```

comment( )

Use this method to set or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $dhcp_ipv6range->comment();
#Modify comment
$dhcp_ipv6range->comment("Modifying the DHCP IPv6 range comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DHCP IPv6 range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value for this field is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $dhcp_ipv6range->disable();
#Modify disable
$dhcp_ipv6range->disable("false");
```

end_addr( )

Use this method to set or retrieve the end address of the DHCP IPv6 range.

An end address is the last IP address in the DHCP IPv6 range available for the clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the end address of the DHCP IPv6 range in lowercase.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get end_addr
my $end_addr = $dhcp_ipv6range->end_addr();
#Modify end_addr
$dhcp_ipv6range->end_addr("2001:db8:0:c000:0021::");
```

exclude( )

Use this method to set or retrieve the exclusion range of IP addresses in the DHCP IPv6 range which should not be assigned to clients.

If static addresses in the middle of a DHCP IPv6 range are assigned to certain hosts, those addresses can be excluded from the DHCP IPv6 range using this method.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::ExclusionRange objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get exclude
my $ref_exclude = $dhcp_ipv6range->exclude();
#Modify exclude
$dhcp_ipv6range->exclude([$exl1]); #$exl1 is an Infoblox::DHCP::ExclusionRange object
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP IPv6 range object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $dhcp_ipv6range->extensible_attributes();
#Modify extensible attributes
$dhcp_ipv6range->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter',
'Tom' ] });
```

ipv6_end_prefix( )

Use this method to set or retrieve the end prefix of the DHCP IPv6 range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the end prefix of the DHCP IPv6 range in lowercase.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_end_prefix
my $ipv6_end_prefix = $dhcp_ipv6range->ipv6_end_prefix();
#Modify ipv6_end_prefix
$dhcp_ipv6range->ipv6_end_prefix("2001:db8:0:c000:0021::");
```

ipv6_prefix_bits( )

Use this method to set or retrieve the prefix bits for the DHCP IPv6 range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 2 and 128.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_prefix_bits
my $ipv6_prefix_bits = $dhcp_ipv6range->ipv6_prefix_bits();
#Modify ipv6_prefix_bits
$dhcp_ipv6range->ipv6_prefix_bits(32);
```

ipv6_start_prefix( )

Use this method to set or retrieve the start prefix of the DHCP IPv6 range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the start prefix of the DHCP IPv6 range in lowercase.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_start_prefix
my $ipv6_start_prefix = $dhcp_ipv6range->ipv6_start_prefix();
#Modify ipv6_start_prefix
$dhcp_ipv6range->ipv6_start_prefix("2001:db8:0:c000:0011::");
```

member( )

Use this method to set or retrieve the member that serves DHCP for this IP address range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

If the attribute is set, the server_association_type attribute will also be set: to "MEMBER" if setting member to a defined value, to "NONE" otherwise.

Parameter

The valid value is a defined Infoblox::DHCP::Member object. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get member
my $ref_member = $dhcp_ipv6range->member();
```

```
#Modify member
$dhcp_ipv6range->member($member1); #$member is an Infoblox::DHCP::Member object
```

name( )

Use this method to set or retrieve a descriptive name for the DHCP IPv6 range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The desired name in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $dhcp_ipv6range->name();
#Modify name
$dhcp_ipv6range->name("external");
```

network( )

Use this method to set or retrieve the DHCP IPv6 network of the address range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DHCP IPv6 Network.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network
my $network = $dhcp_ipv6range->network();
#Set the network
$dhcp_ipv6range->network("2001:db8:0:c000::/51");
```

network_view( )

Use this method to set or retrieve the network view of the DHCP IPv6 range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the DHCP IPv6 range is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $dhcp_ipv6range->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$dhcp_ipv6range->network_view($nview);
```

next_available_ip( )

Use this method to get the next available IPv6 address in the DHCP IPv6 range object. Note that to use this method, the object MUST

be retrieved via a get or search call, or have been created first with an add call.

Parameter

This method does not accept input parameters.

Returns

This method returns the next available IP address in the range in string format.

Within a DHCP Range object, the next available IP address is the first address in this range that matches all of the following rules:

```
   * There is no fixed address or host address record with this IP address
   * There is no unmanaged record with this IP address
   * The IP address is not in any excluded range of this IPv6 DHCP range
   * There is no active lease with this IP address (Deleted leases in the recycle bin are not
considered.)
```

Example

```
# Retrieve a particular range
my @retrieved_objs = $session->get(
                        object     => "Infoblox::DHCP::IPv6Range",
                        start_addr => "10:9::"
);

$range = $retrieved_objs[0];

# Print the next available IPv6 address in the range
print $range->next_available_ip() . "\n";
```

override_recycle_leases( )

The override_recycle_leases attribute controls whether the recycle_leases value in the DHCP IPv6 Range is used, instead of the grid default.

The override_recycle_leases attribute can be specified explicitly. It is also set implicitly when recycle_leases is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for recycle_leases. Set the parameter to "false" to inherit the grid-level setting for recycle_leases.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_recycle_leases
my $override_recycle_leases=$dhcp_ipv6range->override_recycle_leases( );
#Modifying override_recycle_leases
$dhcp_ipv6range->override_recycle_leases("true");
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag of the DHCP IPv6 range.

If the flag is enabled, the leases are kept in recycle bin until one week after expiration. When the flag is disabled, the leases are irrecoverably deleted.

Setting this method to a defined value implicitly sets the override_recycle_leases method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_recycle_leases attribute to "false".

Note that when recycle_leases is set to a defined value and override_recycle_leases is set to "false", the last operation takes precedence. Thus the sequence $object->recycle_leases("true"); $object->override_recycle_leases("false"); will set override_recycle_leases to "false", and the sequence $objct->override_recycle_leases("false"); $object->recycle_leases("true"); will result in override_recycle_leases="true".

Parameter

Specify "true" to set the recycle_leases flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $dhcp_ipv6range->recycle_leases();
#Modify recycle_leases
$dhcp_ipv6range->recycle_leases("false");
```

server_association_type( )

Use this method to set or retrieve the DHCP IPv6 range server association type.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The server_association_type attribute can be specified explicitly. It is also set implicitly when member is set.

Parameter

Valid values are "NONE" and "MEMBER".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get server_association_type
my $server_association_type = $dhcp_ipv6range->server_association_type();
#Modify server_association_type
$dhcp_ipv6range->server_association_type("MEMBER");
```

start_addr( )

Use this method to set or retrieve the start address of the DHCP IPv6 range.

A start address is the first IP address in the DHCP IPv6 range available for the clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the start address of the DHCP IPv6 range in lowercase.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get start_addr
my $start_addr = $dhcp_ipv6range->start_addr();
#Modify start_addr
$dhcp_ipv6range->start_addr("2001:db8:0:c000:0001::");
```

template( )

Use this method to set or retrieve the DHCP IPv6 range template name for a DHCP IPv6 Range object.

Template is the name of a DHCP IPv6 range template which will be used to create the DHCP IPv6 range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

The template parameter can be used to only add a new DHCP IPv6 range and cannot be used during object modify or returned during object get.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get template
my $template = $dhcp_ipv6range->template();
#Modify template
$dhcp_ipv6range->template("Template_Name");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP IPv6 range object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP IPv6 Range object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a DHCP Member object
my $memberdhcp = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2"
);
unless($memberdhcp) {
       die("Construct member failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Member object created successfully\n";

#Create the Network object with the member
my $network = Infoblox::DHCP::IPv6Network->new(
    network => "2001:db8:0:a000::/51",
    comment => "add network",
    members => [ $memberdhcp ]
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
    or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network object added to Infoblox appliance successfully\n";

#Construct exclusion range object
my $exclusion_range = Infoblox::DHCP::ExclusionRange->new(
    start_address => "2001:db8:0:a000:0001::1a",
    end_address   => "2001:db8:0:a000:0001::fa",
    comment       => "Excluded range"
);
```

#Create a DHCP IPv6 Range object

```
my $dhcp_range = Infoblox::DHCP::IPv6Range->new(
                                            start_addr => '2001:db8:0:a000:0001::',
                                            end_addr   => '2001:db8:0:a000:0021::',
```

```
                                                network    => '2001:db8:0:a000::/51',
                                                comment    => 'Random comment',
                                                disable    => 'true',
                                                exclude => [$exclusion_range],
                                                ipv6_start_prefix       => '2001:db8:0:c001:0001::',
                                                ipv6_end_prefix         => '2001:db8:0:c001:0021::',
                                                ipv6_prefix_bits        => 12,
                                                member                  => $memberdhcp,
                                                name                    => 'randomrange',
                                                override_recycle_leases => 'true',
                                                recycle_leases          => 'false',
                                                server_association_type => 'MEMBER',
);
unless($dhcp_range) {
       die("Construct DHCP IPv6 Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6 Range object created successfully\n";

#Add the DHCP IPv6 Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP IPv6 Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Range object added to the network successfully\n";
```

#Create a DHCP IPv6 Range object using a template

```
my $dhcp_range_template = Infoblox::DHCP::IPv6RangeTemplate->new(
        name                   => "custom_range_template",
        offset                 => "10",
        number_of_addresses     => "90",
);
unless($dhcp_range_template) {
       die("Construct DHCP IPv6 Range template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6 Range template object created successfully\n";

#Add the DHCP IPv6 Range template object into the Infoblox appliance through a session
$session->add($dhcp_range_template)
    or die("Add DHCP IPv6 Range template object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Range template object added to the network successfully\n";

my $dhcp_range = Infoblox::DHCP::IPv6Range->new(
        start_addr             => "2001:db8:0:a000:0100::",
        end_addr               => "2001:db8:0:a000:0200::",
        network                => "2001:db8:0:a000::/51",
        template               => "custom_range_template"
);
unless($dhcp_range) {
       die("Construct DHCP IPv6 Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6 Range object created successfully\n";

#Add the DHCP IPv6 Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP IPv6 Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Range object added to the network successfully\n";
```

#Get and modify a DHCP IPv6 Range

```
#Get a DHCP IPv6 Range through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::IPv6Range",
    start_addr => "2001:db8:0:a000:0001::",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP IPv6 Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP IPv6 Range object found at least 1 matching entry\n";
```

```
#Modify one of the attributes of the specified DHCP IPv6 Range
$object->comment ("modified comment");

#Apply the changes
$session->modify($object)
    or die("Modify DHCP IPv6 Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Range object modified successfully \n";
```

#Search a DHCP IPv6 Range

```
# search for DHCP IPv6 range object with start address having last octet as 1
my @retrieved_objs = $session->search(
    object     => "Infoblox::DHCP::IPv6Range",
    network    => "2001:db8:0:a000::/51",
    comment    => "Modified.*"
);

my $object = $retrieved_objs[0];
unless ($object) {
    die("Search DHCP IPv6 range failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DHCP IPv6 range object found at least 1 matching entry\n";
```

#Remove a DHCP IPv6 Range

```
#Get a DHCP IPv6 Range through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::IPv6Range",
    start_addr => "2001:db8:0:a000:0001::",
);

my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP IPv6 Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP IPv6 Range object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove DHCP IPv6 Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Range object removed successfully \n";

#submit the Range Template object for removal
$session->remove($dhcp_range_template)
    or die("Remove DHCP IPv6 Range Tempate failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Range Templateobject removed successfully \n";

#Get a DHCP IPv6 Range through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::IPv6Range",
    start_addr => "2001:db8:0:a000:0100::",
);

my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP IPv6 Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP IPv6 Range object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove DHCP IPv6 Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Range object removed successfully \n";

#Remove network

my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::IPv6Network",
    network    => "2001:db8:0:a000::/51",
);
```

```
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP IPv6 Network failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP IPv6 Network object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove DHCP IPv6 Network failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP IPv6 Network object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::DHCP::View, Infoblox::DHCP::ExclusionRange, Infoblox::DHCP::Member, Infoblox::DHCP::Option, Infoblox::DHCP::IPv6RangeTemplate

---

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::DHCP::IPv6RangeTemplate - DHCP IPv6 RangeTemplate object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::IPv6NetworkTemplate->range_templates( )
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
- METHODS
  - comment( )
  - exclude( )
  - extensible_attributes( )
  - member( )
  - name( )
  - number_of_addresses( )
  - offset( )
  - override_recycle_leases( )
  - recycle_leases( )
  - server_association_type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6RangeTemplate - DHCP IPv6 RangeTemplate object.

DESCRIPTION

A DHCP IPv6 range template contains a set of predefined properties and can be used to create DHCP IPv6 ranges in a quick and consistent way. The DHCP IPv6 range created from a DHCP IPv6 range template will inherit the properties defined in the template.

CONSTRUCTOR

```
 my $ipv6_range_template = Infoblox::DHCP::IPv6RangeTemplate->new(
            name                    => $string,                                  #Required
            number_of_addresses      => $num,                                     #Required
            offset                  => $num,                                     #Required
            comment                 => $string,                                  #Optional / Default is
undefined
            exclude                 => [$ExclusionRange1, $ExclusionRange2,...], #Optional / Default is
empty
            extensible_attributes    => { $string => $string | $num, $string => [ $string | $num, ... ],
... }, #Optional / Default is undefined
            member                  => $Member,                                  #Optional / Default is
undefined
            override_recycle_leases => "true" | "false",                          #Optional / Default is
"false"
            recycle_leases          => "true" | "false" | undef,                  #Optional / Default is
undefined
            server_association_type => "NONE" | "MEMBER",                         #Optional / Default is
"NONE"
  );
```

MODULE METHODS

The following functions can be applied to a DHCP IPv6 range template object.

Infoblox::DHCP::IPv6NetworkTemplate->range_templates( )

Use this function to include a range template in an IPv6 network template on the Infoblox appliance. See Infoblox::DHCP::IPv6NetworkTemplate->range_templates() for parameters and return values.

Example

```
#Construct an IPv6 range template object
my $ipv6_range_template = Infoblox::DHCP::IPv6RangeTemplate->new(
            name                    => "custom_range_template",
            number_of_addresses      => "1000",
            offset                  => "10",
    );

my $template = Infoblox::DHCP::Template->new(
            name                    => "custom_range_template",
    );

#Configure a range template in a network template object
my $responce = $network_template->range_templates([$template]);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that can be applied to a DHCP IPv6 range template object.

Infoblox::Session->add( )

Use this method to add a DHCP IPv6 range template object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $ipv6_range_template = Infoblox::DHCP::IPv6RangeTemplate->new(
        name                    => "custom_range_template",
        offset                  => "20",
        number_of_addresses      => "100",
        comment                 => "add range",
        exclude                 => [$exclusion_range_template],
        member                  => $memberdhcp,
        recycle_leases          => "false",
        override_recycle_leases => "true",
        server_association_type => 'MEMBER',
);

# Submit for addition
my $response = $session->add( $ipv6_range_template );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP IPv6 range template object:

name - Optional. The name of the DHCP IPv6 range template.
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
my @retrieved_objs = $session->get(
                     object => "Infoblox::DHCP::IPv6RangeTemplate",
                     name   => "custom_range_template"
                     );

my @retrieved_objs = $session->get(
                     object                => "Infoblox::DHCP::IPv6RangeTemplate",
                     extensible_attributes => { 'Site' => 'Santa Clara' }
                    );
```

Infoblox::Session->search( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP IPv6 range template object:
```

```
name    - Optional. The name of the DHCP IPv6 range template (regular expression).
comment - Optional. The comment of the DHCP IPv6 range template (regular expression).
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
my @retrieved_objs = $session->search(
                     object  => "Infoblox::DHCP::IPv6RangeTemplate",
                     name    => "custom_range_template",
                     comment => ".*",
       );

my @retrieved_objs = $session->search(
                     object                => "Infoblox::DHCP::IPv6RangeTemplate",
                     extensible_attributes => { 'Site' => 'Santa Clara' }
       );
```

Infoblox::Session->modify( )

Use this method to modify a DHCP IPv6 range template object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment.
$ipv6_range_template->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $ipv6_range_template );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP IPv6 range template object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP IPv6 range template object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::IPv6RangeTemplate",
    name   => "custom_range_template"
    );

# Find the desired object from the retrieved list.
my $desired_ipv6_range_template = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_ipv6_range_template );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP IPv6 range template object.

comment( )

Use this method to set or retrieve a descriptive comment of a DHCP IPv6 range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $ipv6_range_template->comment();
#Modify comment
$ipv6_range_template->comment("Modifying the DHCP IPv6 range template comment");
```

exclude( )

Use this method to set or retrieve the exclusion range template in the DHCP IPv6 range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::ExclusionRangeTemplate objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get exclude
my $ref_exclude = $ipv6_range_template->exclude();
#Modify exclude
$ipv6_range_template->exclude([$exl1]); #$exl1 is an Infoblox::DHCP::ExclusionRangeTemplate object
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP IPv6 range template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $ipv6_range_template->extensible_attributes();
#Modify extensible attributes
$ipv6_range_template->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

member( )

Use this method to set or retrieve the grid member assigned to the DHCP IPv6 range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a defined Infoblox::DHCP::Member object. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get member
my $ref_member = $ipv6_range_template->member();
#Modify member
$ipv6_range_template->member($member1); #$member is an Infoblox::DHCP::Member object
```

name( )

Use this method to set or retrieve the name of the DHCP IPv6 range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the IPv6 range template in string format with a maximum of 64 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $ipv6_range_template->name();
#Modify name
$ipv6_range_template->name("custom_range_template");
```

number_of_addresses( )

Use this method to set or retrieve the number of addresses in this DHCP IPv6 range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of addresses in this DHCP IPv6 range template.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get number_of_addresses
my $number_of_addresses = $ipv6_range_template-> number_of_addresses();
#Modify number_of_addresses
$ipv6_range_template->number_of_addresses("100");
```

offset( )

Use this method to set or retrieve the address offset of this DHCP IPv6 range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The address offset of this DHCP IPv6 range template. The offset is calculated from the start address of the IPv6 network template that includes the DHCP range template. For example, in the IPv6 network 2010:0:0:10::/64, if the offset is set to 10 and number_of_addresses is set to 100, then the start address of the DHCP IPv6 range created from this range template is 2010:0:0::A and the end address is 2010:0:0::6D.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get offset
my $offset = $ipv6_range_template->offset();
#Modify offset
$ipv6_range_template->offset("10");
```

override_recycle_leases( )

The override_recycle_leases attribute controls whether the recycle_leases value in the DHCP IPv6 range template is used, instead of the grid default.

The override_recycle_leases attribute can be specified explicitly. It is also set implicitly when recycle_leases is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for recycle_leases. Set the parameter to "false" to inherit the grid-level setting for recycle_leases.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_recycle_leases
my $override_recycle_leases=$ipv6_range_template->override_recycle_leases( );
#Modifying override_recycle_leases
$ipv6_range_template->override_recycle_leases("true");
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag of the DHCP IPv6 range template.

If the flag is enabled, the leases are kept in the recycle bin until one week after they expire. When the flag is disabled, the leases are irrecoverably deleted.

Setting this method to a defined value implicitly sets the override_recycle_leases method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_recycle_leases attribute to "false".

Note that when recycle_leases is set to a defined value and override_recycle_leases is set to "false", the last operation takes precedence. Thus the sequence $object->recycle_leases("true"); $object->override_recycle_leases("false"); will set override_recycle_leases to "false", and the sequence $objct->override_recycle_leases("false"); $object->recycle_leases("true"); will result in override_recycle_leases="true".

Parameter

Specify "true" to set the recycle_leases flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $ipv6_range_template->recycle_leases();
#Modify recycle_leases
$ipv6_range_template->recycle_leases("false");
```

server_association_type( )

Use this method to set or retrieve the DHCP IPv6 range template server association type.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "NONE" and "MEMBER".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get server_association_type
my $server_association_type = $ipv6_range_template->server_association_type();
#Modify server_association_type
$ipv6_range_template->server_association_type("MEMBER");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP range template object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP IPv6RangeTemplate object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
              master   => "192.168.1.2",      #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a DHCP Member object
my $memberdhcp = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2"
);
unless($memberdhcp) {
       die("Construct member failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Member object created successfully\n";

#Construct exclusion range template object
my $exclusion_range_template = Infoblox::DHCP::ExclusionRangeTemplate->new(
    offset              => "15",
    number_of_addresses => "50",
    comment             => "Excluded range"
);
unless($exclusion_range_template){
       die("Construct exclusion range template object failed: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
 print "Exclusion range template object created successfully\n";
```

#Create a DHCP IPv6 range template object

```
my $ipv6_range_template = Infoblox::DHCP::IPv6RangeTemplate->new(
        name                   => "custom_range_template",
        offset                 => "20",
        number_of_addresses     => "100",
        comment                => "add range",
        exclude                => [$exclusion_range_template],
        member                 => $memberdhcp,
        recycle_leases          => "false",
        override_recycle_leases => "true",
        server_association_type => 'MEMBER',
);
unless($ipv6_range_template) {
       die("Construct DHCP IPv6 range template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP IPv6 range template object created successfully\n";

#Add the DHCP IPv6 range template object into the Infoblox appliance through a session
$session->add($ipv6_range_template)
    or die("Add DHCP IPv6 range template object failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "DHCP IPv6 range template object added to the network successfully\n";
```

#Get and modify a DHCP IPv6 range template

```
#Get a DHCP IPv6 range template through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::IPv6RangeTemplate",
    name       => "custom_range_template",
);
my $object = $retrieved_objs[0];
unless ($object) {
```

```
        die("Get DHCP IPv6 range template failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP IPv6 range template object found at least 1 matching entry\n";

#Modify one of the attributes of the specified DHCP IPv6 range template
$object->comment("modified comment");

#Apply the changes
$session->modify($object)
    or die("Modify DHCP IPv6 range template failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "DHCP IPv6 range template object modified successfully \n";
```

#Remove a DHCP IPv6 range template

```
#Get a DHCP IPv6 range template through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::IPv6RangeTemplate",
    name       => "custom_range_template",
);

my $object = $retrieved_objs[0];
unless ($object) {
        die("Get DHCP IPv6 range template failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP IPv6 range template object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove DHCP IPv6 range template failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "DHCP IPv6 range template object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::IPv6NetworkTemplate, Infoblox::DHCP::RangeTemplate, Infoblox::DHCP::ExclusionRangeTemplate, Infoblox::DHCP::Template, Infoblox::DHCP::Member

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::IPv6SharedNetwork - DHCP IPv6 Shared Network object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - ddns_domainname( )
  - ddns_enable_option_fqdn( )
  - ddns_generate_hostname( )
  - ddns_ttl( )
  - disable( )
  - domain_name( )
  - domain_name_servers( )
  - enable_ddns( )
  - extensible_attributes( )
  - name( )
  - networks( )
  - network_view( )
  - options( )
  - override_ddns_domainname( )
  - override_ddns_enable_option_fqdn( )
  - override_ddns_generate_hostname( )
  - override_ddns_ttl( )
  - override_domain_name( )
  - override_domain_name_servers( )
  - override_enable_ddns( )
  - override_options( )
  - override_preferred_lifetime( )
  - override_update_dns_on_lease_renewal( )
  - override_valid_lifetime( )
  - preferred_lifetime( )
  - update_dns_on_lease_renewal( )
  - valid_lifetime( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::IPv6SharedNetwork - DHCP IPv6 Shared Network object

DESCRIPTION

You can create a shared network when two subnets share a particular network segment. Before creating a shared network, you must first create the subnetworks. For example, you must first create the networks 2001:10:20:: and 2001:10:30:: before designating them as a shared network.

CONSTRUCTOR

```
  my $sharednetwork = Infoblox::DHCP::IPv6SharedNetwork->new(
        name                                => $string,                     #Required
        networks                            => [ $Network1,$Network2,...], #Required
        comment                             => $string | undef,             #Optional / Default is undefined
        ddns_domainname                     => $string | undef,             #Optional / Default is
undefined
        ddns_enable_option_fqdn             => "true" | "false",            #Optional / Default is "false"
        ddns_generate_hostname              => "true" | "false",            #Optional / Default is "false"
        ddns_ttl                            => $num,                        #Optional / Default is 0
        disable                             => "true" | "false",            #Optional / Default is "false"
        domain_name                         => $string | undef,             #Optional / Default is
undefined
```

```
        domain_name_servers                 => "$ipv6addr,$ipv6addr,...",  #Optional / Default is
undefined
        enable_ddns                         => "true" | "false" | undef,   #Optional / Default is
undefined
        extensible_attributes               => { $string => $string | $num, $string => [ $string | $num,
... ], ... }, #Optional / Default is undefined.
        network_view                        => $NetworkView,               #Optional / Default is "default"
network view
        options                             => [$Option1,$Option2,...],    #Optional / Default is empty
list
        override_ddns_domainname            => "true" | "false",           #Optional / Default is "false"
        override_ddns_enable_option_fqdn    => "true" | "false",           #Optional / Default is "false"
        override_ddns_generate_hostname     => "true" | "false",           #Optional / Default is "false"
        override_ddns_ttl                   => "true" | "false",           #Optional / Default is "false"
        override_domain_name                => "true" | "false",           #Optional / Default is "false"
        override_domain_name_servers        => "true" | "false",           #Optional / Default is "false"
        override_enable_ddns                => "true" | "false",           #Optional / Default is "false"
        override_options                    => "true" | "false",           #Optional / Default is "false"
        override_preferred_lifetime         => "true" | "false",           #Optional / Default is "false"
        override_update_dns_on_lease_renewal=> "true" | "false",           #Optional / Default is "false"
        override_valid_lifetime             => "true" | "false",           #Optional / Default is "false"
        preferred_lifetime                  => $num,                       #Optional / Default is 27000
        update_dns_on_lease_renewal         => "true" | "false" | undef,   #Optional / Default is
undefined
        valid_lifetime                      => $num,                       #Optional / Default is 43200
    );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a DHCP IPv6 shared network object.

Infoblox::Session->add( )

Use this method to add an IPv6 shared network object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a network object
my $network1 = Infoblox::DHCP::IPv6Network->new(
      network => "2001:10:20:30::/64",
);

#Construct a shared network object
  my $sharednetwork = Infoblox::DHCP::IPv6SharedNetwork->new(
      name     => "myname",
      networks => [$network1],
);

# Submit for adding a shared network
my $response = $session->add( $sharednetwork );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching shared network objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP shared network object:

 name           - Required.    The name of the shared network in string format.
 network_view   - Optional.    The network view in which the shared network is located. By default,
all network views are searched.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
 my  @result_array = $session->get(
      object       => "Infoblox::DHCP::IPv6SharedNetwork" ,
      name         => "myname",
      network_view => "default"
 );
```

Infoblox::Session->modify( )

Use this method to modify an IPv6 shared network object in the Infoblox appliance. See Infoblox::Session->modify() for parameters

and return values.

Example

```
#Use this method to modify the enable_ddns method.
$object->enable_ddns("true");
# Submit for modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove a shared network object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the object, and then submit it for removal.

Example

```
# Get the object under the same name
my  @result_array = $session->get(
      object       => "Infoblox::DHCP::IPv6SharedNetwork" ,
      name         => "myname",
      network_view => "default"
 );

# Find the desired object on the retrieved list
my $desired_network = $result_array[0];

# Submit for removal
my $response = $session->remove( $desired_network );
```

Infoblox::Session->search( )

Use this method to search for a DHCP shared network object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP shared network object:

 name          - Required. The name of the shared network in string format (regular expression).
 network_view  - Optional. The network view in which the shared network is located. By default, all
network views are searched.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# Search for all DHCP shared networks in the "default" network view
    my @retrieved_objs = $session->search(
      object    => "Infoblox::DHCP::IPv6SharedNetwork",
      name      => "my.*",
      network_view => "default"
    );

# Search for all IPv6 shared networks with the extensible attribute 'Site'
    my @retrieved_objs = $session->search(
      object => "Infoblox::DHCP::IPv6SharedNetwork",
      extensible_attributes => { 'Site' => 'Santa Clara' }
    );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of DHCP a shared network object

comment( )

Use this method to set or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $sharednetwork->comment();
#Modify comment
$sharednetwork->comment("Modified shared network");
```

ddns_domainname( )

Use this method to set or retrieve the ddns_domainname value of a DHCP IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_ddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

The dynamic DNS domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_domainname
my $ddns_domainname = $sharednetwork->ddns_domainname();
#Modify ddns_domainname
$sharednetwork->ddns_domainname("test_domain.com");
```

ddns_enable_option_fqdn( )

Use this method to set or retrieve the ddns_enable_option_fqdn flag of a shared network object. This method controls whether the FQDN option sent by the client is to be used, or if the server can automatically generate the FQDN. This setting overrides the member-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_enable_option_fqdn method to "true". Setting the parameter to undefined causes the appliance to use the member default and automatically resets the override_ddns_enable_option_fqdn attribute to "false".

Note that when ddns_enable_option_fqdn is set to a defined value and override_ddns_enable_option_fqdn is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_enable_option_fqdn("true"); $object->override_ddns_enable_option_fqdn("false"); will set override_ddns_enable_option_fqdn to "false", and the sequence $object->override_ddns_enable_option_fqdn("false"); $object->ddns_enable_option_fqdn("true"); will result in override_ddns_enable_option_fqdn="true".

Parameter

Specify "true" to generate the host name or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute is inherited from the member-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_enable_option_fqdn
my $ddns_enable_option_fqdn = $sharednetwork->ddns_enable_option_fqdn()
# Modify ddns_enable_option_fqdn
$sharednetwork->ddns_enable_option_fqdn("true");
#Un-override
$sharednetwork->ddns_enable_option_fqdn(undef);
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname flag of a shared network object. Enable this method to allow only the DHCP server to generate host name. This setting overrides the member-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_generate_hostname method to "true". Setting the parameter to undefined causes the appliance to use the upper-level setting and automatically resets the override_ddns_generate_hostname attribute to "false".

Note that when ddns_generate_hostname is set to a defined value and override_ddns_generate_hostname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_generate_hostname("true"); $object->override_ddns_generate_hostname("false"); will set override_ddns_generate_hostname to "false", and the sequence $object->override_ddns_generate_hostname("false"); $object->ddns_generate_hostname("true"); will result in override_ddns_generate_hostname="true".

Parameter

Specify "true" to generate the host name or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute is inherited from the member-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_generate_hostname
my $ddns_generate_hostname = $sharednetwork->ddns_generate_hostname()
# Modify ddns_generate_hostname
$sharednetwork->ddns_generate_hostname("true");
#Un-override
$sharednetwork->ddns_generate_hostname(undef);
```

ddns_ttl( )

Use this method to set or retrieve the DNS update Time to Live (TTL) value of an IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_ttl method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_ttl attribute to "false".

Note that when ddns_ttl is set to a defined value and override_dddns_ttl is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_ttl(12); $object->override_ddns_ttl("false"); will set override_ddns_ttl to "false", and the sequence $object->override_ddns_ttl("false"); $object->ddns_ttl(12); will result in override_ddns_ttl="true".

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration,in seconds, that the update is cached. Zero indicates that the update should not be cached. The default value is zero.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get DNS Update TTL
my $dns_ttl = $sharednetwork->ddns_ttl();
#Modify DNS Update TTL
$sharednetwork->ddns_ttl(1200);
```

disable( )

Use this method to set or retrieve the disable flag of an IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag and disable the shared network or "false" to unset the flag and enable the IPv6 shared

network. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get disable
my $disable = $sharednetwork->disable()
# Modify disable
$sharednetwork->disable("true");
```

domain_name( )

Use this method to set or retrieve the domain_name value of a DHCP IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_domain_name attribute to "false".

Note that when domain_name is set to a defined value and override_domain_name is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name("testdomain"); $object->override_domain_name("false"); will set override_domain_name to "false", and the sequence $object->override_domain_name("false"); $object->domain_name("testdomain"); will result in override_domain_name="true".

Parameter

The name of the client's domain for resolving names in DNS in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name
my $domain_name = $sharednetwork->domain_name();
#Modify domain_name
$sharednetwork->domain_name("test_domain.com");
```

domain_name_servers( )

Use this method to set or retrieve the domain_name_servers value of a DHCP IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_domain_name_servers method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_domain_name_servers attribute to "false".

Note that when domain_name_servers is set to a defined value and override_domain_name_servers is set to "false", the last operation takes precedence. Thus the sequence $object->domain_name_servers("2001::1,2001::2"); $object->override_domain_name_servers("false"); will set override_domain_name_servers to "false", and the sequence $object->override_domain_name_servers("false"); $object->domain_name_servers("2001::1,2001::2"); will result in override_domain_name_servers="true".

Parameter

A comma separated list of the domain name server addresses in IPv6 address format. Spaces around the commas are not allowed.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain_name_servers
my $domain_name_servers = $sharednetwork->domain_name_servers();
#Modify domain_name_servers
$sharednetwork->domain_name_servers("2001::1,2001::2");
```

enable_ddns( )

Use this method to set or retrieve the dynamic DNS updates flag of a DHCP IPv6 shared network object. The DHCP server can send DDNS updates to DNS servers in the same grid and to external DNS servers. This setting overrides the member-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_enable_ddns method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_enable_ddns attribute to "false".

Note that when enable_ddns is set to a defined value and override_enable_ddns is set to "false", the last operation takes precedence. Thus the sequence $object->enable_ddns("true"); $object->override_enable_ddns("false"); will set override_enable_ddns to "false", and the sequence $object->override_enable_ddns("false"); $object->enable_ddns("true"); will result in override_enable_ddns="true".

Parameter

Specify "true" to set the dynamic DNS updates flag or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute is inherited from the member-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_ddns
my $enable_ddns = $sharednetwork->enable_ddns()
# Modify enable_ddns
$sharednetwork->enable_ddns("true");
# Un-override enable_ddns
$sharednetwork->enable_ddns(undef);
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with an IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $shared_network->extensible_attributes();
#Modify extensible attributes
$shared_network->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter',
'Tom' ] });
```

name( )

Use this method to retrieve the name of an IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the shared network in string format. A name can have a maximum of 32 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $sharednetwork = $sharednetwork->name();
```

networks( )

Use this method to retrieve the DHCP IPv6 networks associated with an IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::IPv6Network objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get networks
my $sharednetwork = $sharednetwork->networks();
#Modify networks
my $network1 = Infoblox::DHCP::IPv6Network->new(
                                    "network" => "2001:10:20:30::/64",
                                );
my $network2 = Infoblox::DHCP::IPv6Network->new(
                                    "network" => "2001:10:20:40::/64",
                                );
my $sharednetwork = $sharednetwork->networks([$network1, $network2]);
```

network_view( )

Use this method to set or retrieve the network view of the DHCP IPv6 shared network.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the DHCP shared network is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $sharednetwork->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$sharednetwork->network_view($nview);
```

options( )

Use this method to set or retrieve the options value of DHCP IPv6 shared network. DHCP options provide network configuration settings and various services available on the network for DHCP. This setting overrides the member-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_options method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_options attribute to "false".

Note that when the options method is set to a defined value and override_options is set to "false", the last operation takes precedence. Thus the sequence $object->options([$opt]); $object->override_options("false"); will set override_options to "false", and the sequence $object->override_options("false"); $object->options([$opt]); will result in override_options="true".

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $sharednetwork->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name   => "dhcp6.rapid-commit",
    value  => "true"
);
$sharednetwork->options([$option1]);
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$sharednetwork->override_ddns_domainname( );
#Modifying override_ddns_domainname
$sharednetwork->override_ddns_domainname("true");
```

override_ddns_enable_option_fqdn( )

The override_ddns_enable_option_fqdn attribute controls whether the enable_ddns value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_ddns_enable_option_fqdn attribute can be specified explicitly. It is also set implicitly when enable_ddns is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for enable_ddns. Set the parameter to "false" to inherit the grid-level setting for enable_ddns.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_enable_option_fqdn
my $override_ddns_enable_option_fqdn=$sharednetwork->override_ddns_enable_option_fqdn( );
#Modifying override_ddns_enable_option_fqdn
$sharednetwork->override_ddns_enable_option_fqdn("true");
```

override_ddns_generate_hostname( )

The override_ddns_generate_hostname attribute controls whether the enable_ddns value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_ddns_generate_hostname attribute can be specified explicitly. It is also set implicitly when enable_ddns is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for enable_ddns. Set the parameter to "false" to inherit the grid-level setting for enable_ddns.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_generate_hostname
my $override_ddns_generate_hostname=$sharednetwork->override_ddns_generate_hostname( );
#Modifying override_ddns_generate_hostname
$sharednetwork->override_ddns_generate_hostname("true");
```

override_ddns_ttl( )

The override_ddns_ttl attribute controls whether the ddns_ttl value in the DHCP shared network object is used, instead of the grid default.

The override_ddns_ttl attribute can be specified explicitly. It is also set implicitly when ddns_ttl is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_ttl. Set the parameter to "false" to inherit the grid-level setting for ddns_ttl.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_ttl
my $override_ddns_ttl=$sharednetwork->override_ddns_ttl( );
#Modifying override_ddns_ttl
$sharednetwork->override_ddns_ttl("true");
```

override_domain_name( )

The override_domain_name attribute controls whether the domain_name value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_domain_name attribute can be specified explicitly. It is also set implicitly when domain_name is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for domain_name. Set the parameter to "false" to inherit the grid-level setting for domain_name.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_domain_name
my $override_domain_name=$sharednetwork->override_domain_name( );
#Modifying override_domain_name
$sharednetwork->override_domain_name("true");
```

override_domain_name_servers( )

The override_domain_name_servers attribute controls whether the domain_name_servers value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_domain_name_servers attribute can be specified explicitly. It is also set implicitly when domain_name_servers is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for domain_name_servers. Set the parameter to "false" to inherit the grid-level setting for domain_name_servers.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_domain_name_servers
my $override_domain_name_servers=$sharednetwork->override_domain_name_servers( );
#Modifying override_domain_name_servers
$sharednetwork->override_domain_name_servers("true");
```

override_enable_ddns( )

The override_enable_ddns attribute controls whether the enable_ddns value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_enable_ddns attribute can be specified explicitly. It is also set implicitly when enable_ddns is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for enable_ddns. Set the parameter to "false" to inherit the grid-level setting for enable_ddns.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_enable_ddns
my $override_enable_ddns=$sharednetwork->override_enable_ddns( );
#Modifying override_enable_ddns
$sharednetwork->override_enable_ddns("true");
```

override_options( )

The override_options attribute controls whether the options value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_options attribute can be specified explicitly. It is also set implicitly when options is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_options
my $override_options=$sharednetwork->override_options( );
#Modifying override_options
$sharednetwork->override_options("true");
```

override_preferred_lifetime( )

The override_preferred_lifetime attribute controls whether the preferred_lifetime value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_preferred_lifetime attribute can be specified explicitly. It is also set implicitly when preferred_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_preferred_lifetime
my $override_preferred_lifetime=$sharednetwork->override_preferred_lifetime( );
#Modifying override_preferred_lifetime
$sharednetwork->override_preferred_lifetime("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the DHCP shared network object is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$sharednetwork->override_update_dns_on_lease_renewal( );
#Modifying override_update_dns_on_lease_renewal
$sharednetwork->override_update_dns_on_lease_renewal("true");
```

override_valid_lifetime( )

The override_valid_lifetime attribute controls whether the valid_lifetime value in the DHCP IPv6 shared network is used, instead of the grid default.

The override_valid_lifetime attribute can be specified explicitly. It is also set implicitly when valid_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for options. Set the parameter to "false" to inherit the grid-level setting for options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_valid_lifetime
my $override_valid_lifetime=$sharednetwork->override_valid_lifetime( );
#Modifying override_valid_lifetime
$sharednetwork->override_valid_lifetime("true");
```

preferred_lifetime( )

Use this method to set or retrieve the preferred lifetime of the DHCP IPv6 shared network object.

Setting this method to a defined value implicitly sets the override_preferred_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_preferred_lifetime attribute to "false".

Note that when preferred_lifetime is set to a defined value and override_preferred_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->preferred_lifetime(20000); $object->override_preferred_lifetime("false"); will set override_preferred_lifetime to "false", and the sequence $object->override_preferred_lifetime("false"); $object->preferred_lifetime(20000); will result in override_preferred_lifetime="true".

Parameter

Preferred lifetime value in seconds. The default value is 27000. Note that the preferred lifetime cannot be greater than the valid lifetime.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get preferred_lifetime
my $preferred_lifetime = $sharednetwork->preferred_lifetime();
#Modify preferred_lifetime
$sharednetwork->preferred_lifetime(12345);
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$sharednetwork->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$sharednetwork->update_dns_on_lease_renewal("true");
```

valid_lifetime( )

Use this method to set or retrieve the valid lifetime value of a DHCP IPv6 shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_valid_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_valid_lifetime attribute to "false".

Note that when valid_lifetime is set to a defined value and override_valid_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->valid_lifetime(691200); $object->override_valid_lifetime("false"); will set override_valid_lifetime to "false", and the sequence $object->override_valid_lifetime("false"); $object->valid_lifetime(691200); will result in override_valid_lifetime="true".

Parameter

A 32-bit integer (valid range from 1 to 4294967295) that represents the lifetime, in seconds. The default value is 43200.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get valid_lifetime
my $valid_lifetime = $sharednetwork->valid_lifetime();
#Modify valid_lifetime
$sharednetwork->valid_lifetime(864000);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP IPv6 shared network object insertion

```
#Create a session to the Infoblox device
my $session = Infoblox::Session->new(
    master   =>  "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a DHCP Network object>

my $network1 = Infoblox::DHCP::IPv6Network->new(
                                    "network" => "2001:10:20:30::/64",
                                );
my $network2 = Infoblox::DHCP::IPv6Network->new(
                                    "network" => "2001:10:20:40::/64",
                                );
my $response = $session->add($network1)
or die("Add \"2001:10:20:30::/64\"Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network \"2001:10:20:30::/64\"added successfully\n"; my $response = $session->add($network2)
       or die("Add  \"2001:10:20:30::/64\" Network failed: ",
```

```
            $session->status_code() . ":" . $session->status_detail());

print "Network  \"2001:10:20:40::/64\",added successfully\n";
```

#Create a DHCP IPv6 Shared Network object

```
my $sharednetwork1 = Infoblox::DHCP::IPv6SharedNetwork->new(
                                        "name"                          => "myname",
                                        "comment"                       => "add shared network",
                                        "ddns_domainname"               => 'ddns.com',
                                        "ddns_ttl"                      => 123,
                                        "networks"                      => [ $network1 , $network2 ],
                                        "disable"                       => "false",
                                        "domain_name"                   => 'bar.com',
                                        "domain_name_servers"           => '2001::1,2001::2',
                                        "enable_ddns"                   => 'false',
                                        "ddns_generate_hostname"        => 'false',
                                        "override_ddns_domainname"      => 'true',
                                        "override_ddns_ttl"             => 'true',
                                        "override_domain_name"          => 'true',
                                        "override_domain_name_servers"  => 'true',
                                        "override_enable_ddns"          => 'true',
                                        "override_valid_lifetime"       => 'true',
                                        "override_preferred_lifetime"   => 'true',
                                        "override_update_dns_on_lease_renewal" => 'true',
                                        "preferred_lifetime"            => 1234,
                                        "update_dns_on_lease_renewal"   => 'true',
                                        "valid_lifetime"                => 12345,
                                        );

my $response = $session->add( $sharednetwork1 )
or die("Add IPv6 Shared Network failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "IPv6 Shared Network added successfully\n";
```

#Search for a specific IPv6 Shared Network

```
#Search the network
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::IPv6SharedNetwork",
    "name"  => "my.*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search IPv6 Shared Network failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Search IPv6 Shared Network found at least 1 matching entry\n";
```

#Get and modify a IPv6 Shared Network attribute

```
#Get IPv6 Shared Network through the session
my  @retrieved_objs = $session->get(
          object =>"Infoblox::DHCP::IPv6SharedNetwork" ,
          "name" => "myname"
       );

my $object = $retrieved_objs[0]; unless ($object) {
     die("Get IPv6 Shared Network object failed: ",
         $session->status_code() . ":" . $session->status_detail());
 }

print "Get IPv6 Shared Network found at least 1 matching entry\n";

#Modify one of the attributes of the specified Network attribute>
$object->disable("true");
#Apply the changes
my $res = $session->modify($object) ;
if ( !$res)
{
     print "Modify IPv6 Shared Network attribute failed: ".$session->status_code() . ":" . $session-
>status_detail();

}
```

```
print "IPv6 Shared Network object modified successfully \n";
```

#Remove a IPv6 Shared Network Object

```
#Get Network object through the session
my @retrieved_objs = $session->get(
          object  => "Infoblox::DHCP::IPv6SharedNetwork" ,
          name    => "myname"
   );
my $object = $retrieved_objs[0];    unless ($object) {
   die("Get IPv6 Shared Network object failed: ",
       $session->status_code() . ":" . $session->status_detail());
}

print "Get IPv6 Shared Network object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
     or die("Remove IPv6 Shared Network failed: ",
         $session->status_code() . ":" . $session->status_detail());

print "IPv6 Shared Network removed successfully \n";

#Submit the network object for removal
my $response = $session->remove($network1)
or die("Remove \"2001:10:20:30::/64\"Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network \"2001:10:20:30::/64\"removed successfully\n";

#Submit the network object for removal
my $response = $session->remove($network2)
       or die("Remove  \"11.0.0.0\"Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network \"2001:10:20:40::/64\"removed successfully\n";

     ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::Network, Infoblox::DHCP::View, Infoblox::DHCP::Option, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Lease - DHCP Lease object.

- NAME
- DESCRIPTION
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - billing_class( )
  - binding_state( )
  - client_hostname( )
  - cltt( )
  - discovered_name( )
  - discoverer( )
  - ends( )
  - first_discovered( )
  - hardware( )
  - ip_address( )
  - ipv4addr( )
  - ipv6_duid( )
  - ipv6_iaid( )
  - ipv6_preferred_lifetime( )
  - ipv6_prefix_bits( )
  - last_discovered( )
  - mac( )
  - network( )
  - network_component_description( )
  - network_component_ip( )
  - network_component_name( )
  - network_component_port_description( )
  - network_component_port_name( )
  - network_component_port_number( )
  - network_component_type( )
  - network_view( )
  - netbios( )
  - next_binding_state( )
  - on_commit( )
  - on_expiry( )
  - on_release( )
  - option( )
  - os( )
  - port_duplex( )
  - port_link_status( )
  - port_speed( )
  - port_status( )
  - port_vlan_description( )
  - port_vlan_name( )
  - port_vlan_number( )
  - protocol( )
  - starts( )
  - tsfp( )
  - tstp( )
  - uid( )
  - username( )
  - served_by( )
  - variable( )
  - v_cluster( )
  - v_datacenter( )
  - v_host( )
  - v_name( )
  - v_netadapter( )
  - v_switch( )
  - v_type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Lease - DHCP Lease object.

DESCRIPTION

A DHCP lease provides information about the lease of an IP address to a DHCP client, such as the time when the Infoblox appliance issued or freed an IP address, the MAC address and host name of the client that received the IP address, and the grid member that supplied the lease. It allows the Infoblox appliance to store and correlate DHCP lease information over the lifetime of a lease.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that can be applied to a DHCP Lease object.

Infoblox::Session->get( )

Use this method to retrieve all the matching DHCP Lease objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP Lease object:

ip_address    - Optional. IPv4 or IPv6 address of the DHCP lease. Use this, in place of ipv4addr,
even for IPv4 addresses.
ipv4addr      - Optional. IPv4 address (32 bits) of the DHCP lease. This attribute is being
deprecated in favor of the ip_address attribute.
ipv6_duid     - Optional. IPv6 DUID of the DHCP lease. (For IPv6 leases only)
protocol      - Optional. Specify whether to get IPv4, IPv6, or both leases. Allowed values are:
'IPV4', 'IPV6', and 'BOTH'.
start_addr    - Optional. Start IPv4 or IPv6 address of the DHCP range.
end_addr      - Optional. End IPv4 or Ipv6 address of the DHCP range.
network_view  - Optional. Name of the network view that contains the lease.
mac           - Optional. MAC address of the DHCP lease. (For IPv4 leases only)
username      - Optional. User name associated with the DHCP lease.
v_cluster     - Optional. Discovered VMware cluster name. (For IPv4 leases only)
v_datacenter  - Optional. Discovered VMware datacenter name. (For IPv4 leases only)
v_host        - Optional. Discovered VMware server name. (For IPv4 leases only)
v_name        - Optional. Discovered VMware host or virtual machine name. (For IPv4 leases only)
v_netadapter  - Optional. Discovered VMware physical network adapter name. (For IPv4 leases only)
v_switch      - Optional. Discovered VMware virtual switch name. (For IPv4 leases only)
v_type        - Optional. Discovered VMware entity type (possible values: "VirtualMachine",
"HostSystem", "VirtualCenter").
```

Example

```
#Get a DHCP lease by IP address.
my @retrieved_objs = $session->get(
                        object   => "Infoblox::DHCP::Lease",
                        ip_address => "10.0.0.10"
                        mac      => "22:22:22:22:22:22",
                        username => "johndoe"
                        );
my @retrieved_objs = $session->get(
                        object   => "Infoblox::DHCP::Lease",
                        ip_address => "2001::1"
                        username => "johndoe"
                        );

#Get DHCP leases in a DHCP range.
my @retrieved_objs = $session->get(
                        object     => "Infoblox::DHCP::Lease" ,
                        start_addr => "10.0.0.1",
                        end_addr   => "10.0.0.10"
                        );
my @retrieved_objs = $session->get(
                        object     => "Infoblox::DHCP::Lease" ,
                        start_addr => "2001::1",
                        end_addr   => "2001::5"
                        );

#Get DHCP leases by v_type:
my @retrieved_objs = $session->get(
  object    => 'Infoblox::DHCP::Lease',
  v_type    => 'VirtualMachine');
```

Infoblox::Session->remove( )

Use this method to remove a DHCP Lease object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Remove a DHCP IPv4 Lease object.
my $object = $session->get(
    object   => "Infoblox::DHCP::Lease",
    ip_address => "10.0.0.10"
    mac      => "22:22:22:22:22:22",
    username => "johndoe"
);
my $response = $session->remove($object);

#Remove a DHCP IPv6 Lease object.
my $object = $session->get(
    object   => "Infoblox::DHCP::Lease",
    ip_address => "2001::1"
    username => "johndoe"
);
my $response = $session->remove($object);
```

Infoblox::Session->search()

Use this method to search for DHCP Lease objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
  Apply the following attributes to search for a DHCP Lease object:

  ip_address    - Optional. IPv4 or IPv6 address of the DHCP lease. Use this, in place of ipv4addr,
even for IPv4 addresses (regular expression).
  ipv4addr      - Optional. IPv4 address (32 bits) of the DHCP lease. This attribute is being
deprecated in favor of the ip_address attribute (regular exression).
  ipv6_duid     - Optional. IPv6 DUID of the DHCP lease (regular expression). (For IPv6 leases only)
  protocol      - Optional. Specify whether to get IPv4, IPv6, or both leases. Allowed values are:
'IPV4', 'IPV6', and 'BOTH'.
  mac           - Optional. MAC address of the DHCP lease (regular expression). (For IPv4 leases
only)
  network_view  - Optional. Name of the network view that contains the lease.
  username      - Optional. User name associated with the DHCP lease (regular expression).
  start_addr    - Optional. Start IPv4 or IPv6 address of the DHCP range.
  end_addr      - Optional. End IPv4 or IPv6 address of the DHCP range.
  v_cluster     - Optional. Discovered VMware cluster name (regular expression). (For IPv4 leases
only)
  v_datacenter  - Optional. Discovered VMware datacenter name (regular expression). (For IPv4 leases
only)
  v_host        - Optional. Discovered VMware server name (regular expression). (For IPv4 leases
only)
  v_name        - Optional. Discovered VMware host or virtual machine name (regular expression). (For
IPv4 leases only)
  v_netadapter  - Optional. Discovered VMware physical network adapter name (regular expression).
(For IPv4 leases only)
  v_switch      - Optional. Discovered VMware virtual switch name (regular expression). (For IPv4
leases only)
  v_type        - Optional. Discovered VMware entity type (possible values: "VirtualMachine",
"HostSystem", "VirtualCenter"). (For IPv4 leases only)
```

Example

```
#Search a DHCP lease.
my @result_array = $session->search(
                   object   => "Infoblox::DHCP::Lease",
                   ip_address => "^10\.*",
                   mac      => "22:.*",
                   username => "john.*"
                   );
my @result_array = $session->search(
                   object   => "Infoblox::DHCP::Lease",
                   ip_address => "^ff::.*",
                   username => "john.*"
                   );

# search a DHCP lease by v_datacenter:
my @retrieved_objs = $session->search(
   object       => 'Infoblox::DHCP::Lease',
   v_datacenter => 'test.*');
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute value of a DHCP Lease object.

billing_class( )

Use this method to retrieve the billing_class of a DHCP Lease object. This method specifies the class to which this lease is currently billed. This method is for IPv4 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get billing_class
my $billing_class = $lease->billing_class();
```

binding_state( )

Use this method to retrieve the binding state for the current lease. The lease state can be one of the following:

```
 Free     : The lease is available for clients to use.
 Active   : The lease is currently in use by a DHCP client.
 Expired  : The lease was in use, but the DHCP client never renewed it, so it is no longer valid.
 Released : The DHCP client returned the lease to the appliance.
 Abandoned: The Infoblox appliance cannot lease this IP address because the appliance received a response
when it pinged the address.
```

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get binding_state
my $binding_state = $lease->binding_state();
```

client_hostname( )

Use this method to retrieve the client_hostname of a DHCP Lease object. This method specifies the host name that the DHCP client sends to the Infoblox appliance using DHCP option 12.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get client_hostname
my $client_hostname = $lease->client_hostname();
```

cltt( )

Use this method to retrieve the CLTT (Client Last Transaction Time) value of a DHCP Lease object. This method specifies the time of the last transaction with the DHCP client for this lease.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get cltt
my $cltt = $lease->cltt();
```

discovered_name( )

Use this method to retrieve the discovered name of a DHCP Lease object.

Parameter

none

Returns

The method returns the discovered name.

Example

```
#Get discovered_name
my $discovered_name = $lease->discovered_name();
```

discoverer( )

Use this method to retrieve the discoverer of a DHCP Lease object.

Parameter

none

Returns

The method returns the discoverer value.

Example

```
#Get discoverer
my $discoverer = $lease->discoverer();
```

ends( )

Use this method to retrieve the end time value of a DHCP Lease object. This method specifies the day, date, and time that a lease ended.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get ends
my $ends = $lease->ends();
```

first_discovered( )

Use this method to retrieve the date and time when the DHCP Lease object was first discovered.

Parameter

none

Returns

The method returns the date and time when the DHCP Lease object was first discovered.

Example

```
#Get first_discovered
my $first_discovered = $lease->first_discovered();
```

hardware( )

Use this method to retrieve the hardware type of a DHCP Lease object. This method specifies the MAC address of the network interface on which the lease will be used. This method is for IPv4 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get hardware
my $hardware = $lease->hardware();
```

ip_address( )

Use this method to retrieve the IPv4 or IPv6 address that the Infoblox appliance leased to the DHCP client. Use this method, in place of the `ipv4addr()` method, even for IPv4 addresses.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get ip_address
my $ip_address = $lease->ip_address();
```

ipv4addr( )

Use this method to retrieve the IP address that the Infoblox appliance leased to the DHCP client. This method is being deprecated in favor of the `ip_address()` method. This method is for IPv4 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $lease->ipv4addr();
```

ipv6_duid( )

Use this method to retrieve the DUID value of an IPv6 address that the Infoblox appliance leased to the DHCP client. This method is for IPv6 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get ipv6_duid
my $ipv6_duid = $lease->ipv6_duid();
```

ipv6_iaid( )

Use this method to retrieve the Interface ID value of an IPv6 address that the Infoblox appliance leased to the DHCP client. This method is for IPv6 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get ipv6_iaid
my $ipv6_iaid = $lease->ipv6_iaid();
```

ipv6_preferred_lifetime( )

Use this method to retrieve the preferred lifetime value of an IPv6 address that the Infoblox appliance leased to the DHCP client. This method is for IPv6 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get ipv6_preferred_lifetime
my $ipv6_preferred_lifetime = $lease->ipv6_preferred_lifetime();
```

ipv6_prefix_bits( )

Use this method to retrieve the prefix bits value of an IPv6 address that the Infoblox appliance leased to the DHCP client. This method is for IPv6 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get ipv6_prefix_bits
my $ipv6_prefix_bits = $lease->ipv6_prefix_bits();
```

last_discovered( )

Use this method to retrieve the time when this object was last discovered.

The default value for this field is empty.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get last_discovered
my $last_discovered = $lease->last_discovered();
```

mac( )

Use this method to retrieve the MAC address of the DHCP client that leased the IP address. This method is for IPv4 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get mac
my $mac = $lease->mac();
```

network( )

Use this method to find the network associated with this lease. This property is read-only.

Parameter

N/A

Returns

The method returns the network associated with the lease in the string of the form "network/netmask".

Example

```
#Get network property of the lease
my $network = $lease->network();
```

network_component_description( )

Use this method to retrieve the description of the network device that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the network device description.

Example

```
#Get network_component_description
my $network_component_description = $lease->network_component_description();
```

network_component_ip( )

Use this method to retrieve the IP address of the network device that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the network device IP address.

Example

```
#Get network_component_ip
my $network_component_ip = $lease->network_component_ip();
```

network_component_name( )

Use this method to retrieve the name of the network device that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the network device name.

Example

```
#Get network_component_name
my $network_component_name = $lease->network_component_name();
```

network_component_port_description( )

Use this method to retrieve the description of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the description of the network device port.

Example

```
#Get network_component_port_description
my $network_component_port_description = $lease->network_component_port_description();
```

network_component_port_name( )

Use this method to retrieve the name of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the name of the port on the network device.

Example

```
#Get network_component_port_name
my $network_component_port_name = $lease->network_component_port_name();
```

network_component_port_number( )

Use this method to retrieve the number of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the network device port number in the range 0-9999.

Example

```
#Get network_component_port_number
my $network_component_port_number = $lease->network_component_port_number();
```

network_component_type( )

Use this method to retrieve the type of the network device that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the network device type.

Example

```
#Get network_component_type
my $network_component_type = $lease->network_component_type();
```

network_view( )

Use this method to retrieve the network view of the DHCP lease.

Parameter

N/A

Returns

This method returns the network view of the DHCP lease as Infoblox::DHCP::View object.

Example

```
#Get network_view
my $nview = $lease->network_view();
```

netbios( )

Use this method to retrieve the name in the NetBIOS reply that responded to a NetBIOS query.

Returns

The method returns the attribute value.

Example

```
#Get netbios
my $netbios = $lease->netbios();
```

next_binding_state( )

Use this method to retrieve the subsequent binding state when the current lease expires. This method is for IPv4 leases only.

```
 Free     : The lease is available for clients to use.
 Active   : The lease is currently in use by a DHCP client.
 Expired  : The lease was in use, but the DHCP client never renewed it, so it is no longer valid.
 Released : The DHCP client returned the lease to the appliance.
 Abandoned: The Infoblox appliance cannot lease this IP address because the appliance received a response
when it pinged the address.
```

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get next_binding_state
my $next_binding_state = $lease->next_binding_state();
```

on_commit( )

Use this method to retrieve the list of commands to be executed upon the granting of the lease.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get on_commit
my $on_commit = $lease->on_commit();
```

on_expiry( )

Use this method to retrieve the list of commands to be executed upon the expiration of the lease.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get on_expiry
my $on_expiry = $lease->on_expiry();
```

on_release( )

Use this method to retrieve the list of commands to be executed upon the release of the lease.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get on_release
my $on_release = $lease->on_release();
```

option( )

Use this method to retrieve the option value of a DHCP Lease object. This method specifies the agent circuit ID and remote ID sent by a DHCP relay agent in DHCP option 82. This method is for IPv4 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get option
my $option = $lease->option();
```

os( )

Use this method to retrieve the operating system associated with this DHCP lease.

Returns

The method returns the attribute value.

Example

```
#Get os
my $os = $lease->os();
```

port_duplex( )

Use this method to retrieve the duplex setting of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the duplex setting of the network device port. The returend value, if any, is one of the following:

```
Full
Half
```

Example

```
#Get port_duplex
my $port_duplex = $lease->port_duplex();
```

port_link_status( )

The method returns the link status of the network device port. The returned value, if any, is one of the following:

Parameter

none

Returns

The method returns the port link status. Returned values, if any, are one of the following:

```
Connected
Not Connected
Unknown
```

Example

```
#Get port_link_status
my $port_link_status = $lease->port_link_status();
```

port_speed( )

Use this method to retrieve the speed of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the speed value of the network device port. The returned value, if any, is one of the following:

```
10M
100M
1G
10G
100G
Unknown
```

Example

```
#Get port_speed
my $port_speed = $lease->port_speed();
```

port_status( )

Use this method to retrieve the status of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the status of the network device port. The returned value, if any, is one of the following:

```
Up
Down
Unknown
```

Example

```
#Get port_status
my $port_status = $lease->port_status();
```

port_vlan_description( )

Use this method to retrieve the VLAN description of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the VLAN description of the network device port.

Example

```
#Get port_vlan_description
my $port_vlan_description = $lease->port_vlan_description();
```

port_vlan_name( )

Use this method to retrieve the VLAN name of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the VLAN name of the network device port.

Example

```
#Get port_vlan_name
my $port_vlan_name = $lease->port_vlan_name();
```

port_vlan_number( )

Use this method to retrieve the VLAN number of the network device port that is connected to the DHCP Lease object.

Parameter

none

Returns

The method returns the VLAN number of the network device port in the range 0-9999.

Example

```
#Get port_vlan_number
my $port_vlan_number = $lease->port_vlan_number();
```

protocol( )

Use this method to determine whether the IP address the Infoblox appliance leased an IPv4 or IPv6 address to the client.

Parameter

N/A

Returns

The method returns the attribute value ('IPV4' or 'IPV6').

Example

```
#Get protocol
my $protocol = $lease->protocol();
```

starts( )

Use this method to retrieve the start time value of a DHCP Lease object. This method specifies the day, date, and time that the lease started.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get starts
my $starts = $lease->starts();
```

tsfp( )

Use this method to retrieve the TSFP (Time Sent From Partner) value of a DHCP Lease object. This method specifies the time that the current lease state ends, from the point of view of a remote DHCP failover peer. This method is for IPv4 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get tsfp
my $tsfp = $lease->tsfp();
```

tstp( )

Use this method to retrieve the TSTP (Time Sent To Partner) value of a DHCP Lease object. This method specifies the time that the

current lease state ends, from the point of view of a local DHCP failover peer. This method is for IPv4 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get tstp
my $tstp = $lease->tstp();
```

uid( )

Use this method to retrieve the UID (User ID) value of a DHCP Lease object. This method specifies the client identifier that the DHCP client sends the Infoblox appliance (in DHCP option 61) when it acquires the lease. Not all DHCP clients send a UID. This method is for IPv4 leases only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get uid
my $uid = $lease->uid();
```

username( )

Use this method to retrieve the user name that the server has associated with a DHCP Lease object.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get username
my $username = $lease->username();
```

served_by( )

Use this method to retrieve the IP address of the server that hands out an active lease to a client.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get served_by
my $served_by = $lease->served_by();
```

variable( )

Use this method to retrieve the variable value of a DHCP Lease object. This method keeps all variables related to the DDNS update of the DHCP lease. The variables can be one of the following:

```
 ddns-text        : The ddns-text variable is used to record the value of the client's TXT identification
record when the
```

```
                    interim DDNS update style has been used to update the DNS service for a particular
lease.

 ddns-fwd-name    : The ddns-fwd-name variable records the value of the name used when the client's A
record was updated, if a DDNS
                    update was successfully completed by the server. The server may have used this name
when it updated the
                    client's PTR record as well.

 ddns-client-fqdn : If the server is configured to use the interim DDNS update style, and is also
configured to allow clients
                    to update their own FQDNs, and the client did in fact update its own FQDN, then the
ddns-client-fqdn
                    variable records the name that the client indicated it used. This is also the name that
the server
                    used to update the client's PTR record in this case.

 ddns-rev-name    : If the server successfully updates the client's PTR record, this variable will record
the name that the
                    DHCP server used for the PTR record. The name to which the PTR record points will be
either the
                    ddns-fwd-name or the ddns-client-fqdn.
```

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get variable
my $variable = $lease->variable();
```

v_cluster( )

Use this method to retrieve the name of the VMware cluster associated with the DHCP Lease object. This method is read-only. This method is for IPv4 leases only.

Parameter

none

Returns

This method returns a string that contains the VMware cluster name. The default value is an empty string.

Example

```
#Get v_cluster
my $v_cluster = $lease->v_cluster();
```

v_datacenter( )

Use this method to retrieve the name of the VMware datacenter associated with the DHCP Lease object. This method is read-only. This method is for IPv4 leases only.

Parameter

none

Returns

This method returns a string that contains the VMware datacenter name. The default value is an empty string.

Example

```
#Get v_datacenter
my $v_datacenter = $lease->v_datacenter();
```

v_host( )

Use this method to retrieve the name of the VMware host associated with the DHCP Lease object. This method is read-only. This method is for IPv4 leases only.

Parameter

none

Returns

This method returns a string that contains the VMware host name. The default value is an empty string.

Example

```
#Get v_host
my $v_host = $lease->v_host();
```

v_name( )

Use this method to retrieve the name of the VMware entity associated with the DHCP Lease object. This method is read-only. This method is for IPv4 leases only.

Parameter

none

Returns

This method returns a string that contains the VMware entity name. The default value is an empty string.

Example

```
#Get v_name
my $v_name = $lease->v_name();
```

v_netadapter( )

Use this method to retrieve the name of the physical network adapter through which the VMware entity is connected to the appliance. This method is read-only. This method is for IPv4 leases ohly.

Parameter

none

Returns

This method returns a string that contains the network adapter name. The default value is an empty string.

Example

```
#Get v_netadapter
my $v_netadapter = $lease->v_netadapter();
```

v_switch( )

Use this method to retrieve the name of the virtual switch through which the VMware entity is connected to the appliance. This method is read-only. This method is for IPv4 leases only.

Parameter

none

Returns

This method returns a string that contains the virtual switch name. The default value is an empty string.

Example

```
#Get v_switch
my $v_switch = $lease->v_switch();
```

v_type( )

Use this method to retrieve the type of VMware entity associated with the DHCP Lease object. This method is read-only. This method is for IPv4 leases only.

Parameter

none

Returns

This method returns a string that contains the VMware entity name. The default value is an empty string. The returned value, if any, can be one of the following:

```
VirtualMachine
HostSystem
VirtualCenter
```

Example

```
#Get v_type
my $v_type = $lease->v_type();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP Lease object, such as get, search, and remove. Also, this sample code includes error handling for the operations.

#Preparation prior to DHCP Lease operations

```
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $host_ip   = "192.168.1.2";
my $host_name = "infoblox.localdomain";
my $session = Infoblox::Session->new(
    master   => $host_ip,       #appliance host ip
    username => "admin",        #appliance user login
    password => "infoblox"      #appliance password
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully.\n";

#Create a member object
 my $member = Infoblox::DHCP::Member->new(
    name     => $host_name,
    ipv4addr => $host_ip,
);

unless($member) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create a network object
my $network = Infoblox::DHCP::Network->new(
                           network  => "10.0.0.0/8",
                           comment  => "add network",
                           members  => [$member]
                           );
unless($network){
       die("Construct network object fail: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
       }

print"Network object created successfully.\n";

#Verify if the network exists
my $object = $session->get(
               object  => "Infoblox::DHCP::Network",
               network => "10.0.0.0/8"
               );
unless($object){
       print"Network does not exist on server, safe to add the network.\n";
       $session->add($network)
               or die("Add network failed: ",
                       $session->status_code(). ":" .$session->status_detail());
       }

print"Network added successfully.\n";

#Create a dhcp range object
my $dhcp_range = Infoblox::DHCP::Range->new(
    network    => "10.0.0.0/8",
    start_addr => "10.0.0.1",
    end_addr   => "10.0.0.10",
    member     => $member,
);
```

```
unless($dhcp_range){
       die("Construct dhcp range object failed: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
       }

print "dhcp range object created successfully.\n";

#Add the dhcp range to the Infoblox appliance
$session->add($dhcp_range)
               or die("Add dhcp range failed: ",
                       $session->status_code(). ":" .$session->status_detail());

print "dhcp range added successfully.\n";

my $result = $session->import_data(
       "type" => "leases",
       "path"=> "./API_unit_test_sample_code/import_dhcpd.lease",
       "format" => "ISC",
       "lease_precedence" => "replace-previous"
      );

unless($result){
        die("Import lease failed: ",
          $session->status_code() . ":" . $session->status_detail());
 }

print "Import dhcp lease successfull.\n";
```

#Search for a DHCP Lease object

```
my @retrieved_objs= $session->search(
                    object   => "Infoblox::DHCP::Lease" ,
                    ipv4addr => '^10\..*'
                    );
my $desired_lease = $retrieved_objs[0];
unless($desired_lease){
        die("search lease failed: ",
          $session->status_code() . ":" . $session->status_detail());
 }

print "Search dhcp lease object found at least 1 matching entry\n";
```

#Get a DHCP Lease object

```
#Get a DHCP lease by IP address.
my @retrieved_objs = $session->get(
                         object   => "Infoblox::DHCP::Lease",
                         ipv4addr => "10.0.0.10"
                         );
my $desired_lease = $retrieved_objs[0];
unless($desired_lease){
        die("get lease failed: ",
          $session->status_code() . ":" . $session->status_detail());
 }

print "Get dhcp lease object found at least 1 matching entry\n";
```

#Remove a DHCP Lease object

```
#Submit the object for removal
$session->remove($desired_lease)
    or die("Remove dhcp lease failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "dhcp lease object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->remove(), Infoblox::Session->search(),

Infoblox::DHCP::Network, Infoblox::DHCP::View, Infoblox::DHCP::Range, Infoblox::DHCP::IPv6Network, Infoblox::DHCP::IPv6Range

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::MAC - DHCP MAC object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - authentication_time( )
  - comment( )
  - expired( )
  - expiration_time( )
  - expiration_type( )
  - extensible_attributes( )
  - guest_custom_field1 ( )
  - guest_custom_field2 ( )
  - guest_custom_field3 ( )
  - guest_custom_field4 ( )
  - guest_email ( )
  - guest_first_name ( )
  - guest_last_name ( )
  - guest_middle_name ( )
  - guest_phone ( )
  - filter( )
  - mac( )
  - is_registered_user( )
  - reserved_for_infoblox( )
  - username( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::MAC - DHCP MAC object.

DESCRIPTION

A MAC Address is a unique hardware identifier. The Infoblox device can filter address requests by the MAC address of a requesting hosts. The filter instructs the Infoblox device either to grant or deny an address request if the requesting host matches the filter. Also, it is possible configure specific MAC addresses within a filter to expire after a certain amount of time has passed.

CONSTRUCTOR

```
 my $mac = Infoblox::DHCP::MAC->new(
       filter                => $string,        #Required
       mac                   => $mac_addr,      #Required
       authentication_time   => $num,           #Optional / Default is empty
       comment               => $string,        #Optional / Default is empty
       expiration_time       => $num,           #Optional / Default is 0 for never expire
       extensible_attributes => { $string => $string | $num, $string => [ $string | $num, ... ], ... },
#Optional / Default is undefined
       guest_custom_field1   => $string,        #Optional / Default is empty
       guest_custom_field2   => $string,        #Optional / Default is empty
       guest_custom_field3   => $string,        #Optional / Default is empty
       guest_custom_field4   => $string,        #Optional / Default is empty
       guest_email           => $string,        #Optional / Default is empty
       guest_first_name      => $string,        #Optional / Default is empty
       guest_last_name       => $string,        #Optional / Default is empty
       guest_middle_name     => $string,        #Optional / Default is empty
       guest_phone           => $string,        #Optional / Default is empty
       is_registered_user    => "true" | "false", #Optional / Default is "true"
       reserved_for_infoblox => $string,        #Optional / Default is "false"
       username              => $string,        #Optional / Default is empty
```

```
);
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DHCP MAC object.

Infoblox::Session->add( )

Use this method to add a DHCP MAC object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a DHCP MAC object
my $mac = Infoblox::DHCP::MAC->new(
          "mac"                  => "00:00:00:00:00:03",
          "filter"               => "new",
          "comment"              => "dhcp mac object",
          "expiration_time"      => "0",
          "username"             => "infoblox_user",
          "authentication_time"  => "0",
          "reserved_for_infoblox" => "For Infoblox Use only",
);
#Submit for addition
my $response = $session->add($mac)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP MAC object:

 mac    - Optional. MAC address of the DHCP MAC object.
 filter - Optional. MAC filter name.
 comment  - Optional. A comment in string format.
 username - Optional. Username of the DHCP MAC object.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
#Get DHCP MAC object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::MAC",
    mac    => "00:00:00:00:00:03",
    filter => "new",
);

# get all objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object               => "Infoblox::DHCP::MAC",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP MAC object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the expiration time.
$mac->expiration_time("1234567890");
#Submit modification
my $response = $session->modify( $mac );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP MAC object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP MAC object, and then submit this object for removal.

Example

```
#Get DHCP MAC object through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::MAC",
    mac        => "00:00:00:00:00:03",
    filter     => "new",
);
#Find the desired object from the retrieved list.
my $desired_mac = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_mac );
```

Infoblox::Session->search( )

Use this method to search for DHCP MAC objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP MAC Address object:

 comment  - Optional. A comment in string format (regular expression).
 mac      - Optional. MAC address of the DHCP MAC object (regular expression).
 username - Optional. Username of the DHCP MAC object (regular expression).
 filter   - Optional. MAC address filter name.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DHCP MAC objects that are specified in the "new" MAC filter.
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::MAC",
    mac    => "00:00:00:00:00.*",
    filter => "new"
);

# search all DHCP MAC associations with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DHCP::MAC",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a DHCP MAC object.

authentication_time( )

Use this method to set or retrieve the authentication time. Indicates the time when the MAC address was first authenticated.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The absolute UNIX time (seconds since UNIX epoch) when the address was authenticated. The maximum value can extend upto 4294967295.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authentication_time
my $authentication_time = $mac->authentication_time();
#Modify authentication_time
$mac->authentication_time("1234567890");
```

comment( )

Use this method to set or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $mac->comment();
#Modify comment
$mac->comment("Modified DHCP MAC comment");
```

expired( )

Use this method to retrieve the expired flag of the DHCP MAC object .

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get expired
my $expired = $mac->expired();
```

expiration_time( )

Use this method to set or retrieve the expiration time of the DHCP MAC object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

By default, the MAC address never expires; otherwise, it is the absolute UNIX time (seconds since UNIX epoch) when the address expires. The maximum value can extend upto 4294967295.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get expiration_time
my $expiration_time = $mac->expiration_time();
#Modify expiration_time
$mac->expiration_time("1234567890");
```

expiration_type( )

Use this method to retrieve the expiration type of the DHCP MAC object.

Omit the parameter to retrieve the attribute value.

Parameter

None.

Returns

If you did not specify a parameter, the method returns the attribute value.

If the return value is "0" means, the MAC Address never expires. If the return value is "1", MAC Address will automatically expire after the time defined in expiration time.

Example

```
#Get expiration_type
my $expiration_type = $mac->expiration_type();
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP MAC object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $mac->extensible_attributes();
#Modify extensible attributes
$mac->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

guest_custom_field1 ( )

Use this method to set or retrieve the value in Custom Field 1 of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field1
my $guest_custom_field1 = $dhcp->guest_custom_field1();
#Modifying guest_custom_field1
$dhcp->guest_custom_field1('Custom value 1');
```

guest_custom_field2 ( )

Use this method to set or retrieve the value in Custom Field 2 of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field2
my $guest_custom_field2 = $dhcp->guest_custom_field2();
#Modifying guest_custom_field2
$dhcp->guest_custom_field2('Custom value 2');
```

guest_custom_field3 ( )

Use this method to set or retrieve the value in Custom Field 3 of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field3
my $guest_custom_field3 = $dhcp->guest_custom_field3();
#Modifying guest_custom_field3
$dhcp->guest_custom_field3('Custom value 3');
```

guest_custom_field4 ( )

Use this method to set or retrieve the value in Custom Field 4 of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field4
my $guest_custom_field4 = $dhcp->guest_custom_field4();
#Modifying guest_custom_field4
$dhcp->guest_custom_field4('Custom value 4');
```

guest_email ( )

Use this method to set or retrieve the email address in the Email field of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_email
my $guest_email = $dhcp->guest_email();
#Modifying guest_email
$dhcp->guest_email('user@domain.com');
```

guest_first_name ( )

Use this method to set or retrieve the entry in the First Name field of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_first_name
my $guest_first_name = $dhcp->guest_first_name();
#Modifying guest_first_name
$dhcp->guest_first_name('John');
```

guest_last_name ( )

Use this method to set or retrieve the entry in the Last Name field of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_last_name
my $guest_last_name = $dhcp->guest_last_name();
#Modifying guest_last_name
$dhcp->guest_last_name('Doe');
```

guest_middle_name ( )

Use this method to set or retrieve the entry in the Middle Name field of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_middle_name
my $guest_middle_name = $dhcp->guest_middle_name();
#Modifying guest_middle_name
$dhcp->guest_middle_name('Robert');
```

guest_phone ( )

Use this method to set or retrieve the entry in the Phone field of the guest registration page of this DHCP MAC object.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_phone
my $guest_phone = $dhcp->guest_phone();
#Modifying guest_phone
$dhcp->guest_phone('1-555-555-5555');
```

filter( )

Use this method to set or retrieve the filter value of the DHCP MAC object .

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the DHCP MAC filter. Maximum length upto 1024 bytes is supported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get filter
my $filter = $mac->filter();
#Modify filter
$mac->filter("filter1");
```

mac( )

Use this method to set or retrieve the MAC address of the DHCP MAC object .

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

MAC Address of the DHCP MAC object. Format for the hexadecimal strings for MAC addresses and vendor prefixes with colons or dashes. Both of the following formats are acceptable: 11:11:11:11:11:11 and 11-11-11-11-11-11.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mac
my $mac_value = $mac->mac();
#Modify mac
$mac->mac("00:00:00:11:11:11");
```

is_registered_user( )

Use this method to set or retrieve the is_registered_user flag of the DHCP MAC object. If this flag is "false", then this DHCP MAC object represents a "Guest" registration.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the is_registered_user flag or "false" to deactivate/unset it. The default value for this field is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get is_registered_user
my $is_registered_user = $mac->is_registered_user()
# Modify is_registered_user
$mac->is_registered_user("true");
```

reserved_for_infoblox( )

Set this flag to reserve the MAC address for an Infoblox appliance.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Reserved to write comment related to this particular MAC address. Comment cannot exceed 1024 bytes long.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get reserved_for_infoblox
my $reserved_for_infoblox = $mac->reserved_for_infoblox();
#Modify reserved_for_infoblox
$mac->reserved_for_infoblox("reserved for Infoblox");
```

username( )

Use this method to set or retrieve the user name of the DHCP MAC object .

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

User name of the DHCP MAC object. Used for authenticated DHCP purposes. Maximum of 128 characters supported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get username
my $username = $mac->username();
#Modify username
$mac->username("user_infoblox");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP MAC object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a DHCP MAC Filter object
my $macfilter = Infoblox::DHCP::Filter::MAC->new ( "name" => "mac_filter");

unless($macfilter) {
     die("Construct mac filter failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "MAC Filter object created successfully\n";

#Verify if the DHCP MAC Filter exists in the Infoblox appliance
my $object = $session->get(object => "Infoblox::DHCP::Filter::MAC", name => "mac_filter");
unless ($object) {
   print "mac filter does not exist on server, safe to add the mac filter\n";
   $session->add($macfilter)
      or die("Add mac filter failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "DHCP MAC Filter added successfully\n";
```

#Create a DHCP MAC object

```
my $mac = Infoblox::DHCP::MAC->new(
           "mac"                  => "00:00:00:00:00:01",
           "filter"               => "mac_filter",
           "comment"              => "dhcp mac object",
           "expiration_time"       => "0",
           "username"             => "infoblox_user",
           "authentication_time"   => "0",
           "reserved_for_infoblox" => "For Infoblox Use only",
);


unless($mac) {
            die("Construct MAC object failed: ",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
}
```

```
print "MAC Address object created successfully\n";

#Add the MAC Address object into the Infoblox appliance through a session
$session->add($mac)
     or die("Add MAC Address object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DHCP MAC Address object added successfully\n";
```

#Search for a specific DHCP MAC Address object

```
#Search for all DHCP MAC address objects
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::MAC",
    filter => "mac_filter" );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search for MAC Address object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search MAC Address object found at least 1 matching entry\n";
```

#Get and modify a MAC Address object

```
#Get MAC Address object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::MAC",
    mac    => "00:00:00:00:00:01",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get MAC Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get MAC Address object found at least 1 matching entry\n";

#Modify one of the attributes of the obtained MAC Address object
$object->expiration_time("1234567890");

#Apply the changes
$session->modify($object)
    or die("Modify MAC Address object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "MAC Address object modified successfully \n";
```

#Remove a MAC Address object

```
#Get MAC Address object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::MAC",
    mac    => "00:00:00:00:00:01",
    filter => "mac_filter",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get MAC Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get MAC Address object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove MAC Address object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "MAC Address object removed successfully \n";

#Submit the MAC Address Filter object for removal
$session->remove($macfilter)
    or die("Remove MAC Address Filter object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "MAC Address Filter object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::Filter::MAC, Infoblox::DHCP::Filter::RelayAgent, Infoblox::DHCP::Lease, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Member - DHCP Member object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::Network->members( )
  - Infoblox::DHCP::Range->member( )
- METHODS
  - ipv4addr( )
  - name( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Member - DHCP Member object.

DESCRIPTION

A grid member is part of a grid which is a group of two or more Infoblox appliances that share sections of a common, distributed, built-in database. A grid member inherits its settings from the grid. A member is a single Infoblox appliance or an HA pair that provides DHCP services seamlessly across an entire network.

CONSTRUCTOR

```
my $member = Infoblox::DHCP::Member->new(
    ipv4addr => $ipv4addr, #Required
    name     => $fqdn,     #Optional
);
```

MODULE METHODS

This section describes all the functions that you can apply to an DHCP Member object.

Infoblox::DHCP::Network->members( )

Use this function to specify the primary member for a network. See Infoblox::DHCP::Network->members() for parameters and return values.

Example

```
my $member1 = Infoblox::DHCP::Member->new(
    name     => "localhost.localdomain",
    ipv4addr => "192.168.1.2",
);

unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create the Network object with this member
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/255.0.0.0",
    members => [$member1],
);
```

Infoblox::DHCP::Range->member( )

Use this function to specify the primary member for a DHCP range. See Infoblox::DHCP::Range->members() for parameters and return values.

Example

```
my $member = Infoblox::DHCP::Member->new(
    name     => "localhost.localdomain",
    ipv4addr => "192.168.1.2",
);

unless($member) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create the DHCP Range object with this member
my $dhcp_range = Infoblox::DHCP::Range->new(
    network    => "10.0.0.0/8",
    start_addr => "10.0.0.1",
    end_addr   => "10.0.0.10",
    member     => $member,
);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a DHCP Member object.

ipv4addr( )

Use this method to set or retrieve the IPv4 address of the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

IPv4 address (32 bits) of the member.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4Addr
my $ipv4Addr = $member->ipv4Addr();
#Modify ipv4Addr
$member->ipv4Addr("192.168.1.3");
```

name( )

Use this method to set or retrieve the name of the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A host name can have a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $member->name();
#Modify name
$member->name("dhcp.infoblox.com");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP Member object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP Member object

```
my $member1 = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
);

unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create the Network object with this member
my $network = Infoblox::DHCP::Network->new(
                   network => "10.0.0.0/255.0.0.0",
                   comment => "add network",
                   members => [ $member1 ]
);

unless($network) {
            die("Construct Network object failed: ",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
     or die("Add Network object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DHCP member object added to the network successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::Network, Infoblox::Session->add(),Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::MSOption - Microsoft Server DHCP Option object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::MSServer::DHCP->ms_options( )
  - Infoblox::DHCP::Range->ms_options( )
  - Infoblox::DHCP::FixedAddr->options( )
- METHODS
  - name( )
  - code( )
  - type( )
  - value( )
  - user_class( )
  - vendor_class( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::MSOption - Microsoft (r) Server DHCP Option object.

DESCRIPTION

DHCP options describe network configuration settings and various services available on the network. These options occur as variable-length fields at the end of DHCP messages.

You can add or change only DHCP options that are in the Microsoft server and NIOS predefined list of options. Options that are not common to both are read-only.

Following are the options that the Microsoft server and the NIOS predefined list of options have in common:

```
 code    name
 ========================================================
   2    'Time Offset'
   4    'Time Server'
   5    'Name Servers'
   7    'Log Servers'
   8    'Cookie Servers'
   9    'LPR Servers'
   10   'Impress Servers'
   11   'Resource Location Servers'
   12   'Host Name'
   13   'Boot File Size'
   14   'Merit Dump File'
   16   'Swap Server'
   17   'Root Path'
   18   'Extensions Path'
   19   'IP Layer Forwarding'
   20   'Nonlocal Source Routing'
   21   'Policy Filter Masks'
   22   'Max DG Reassembly Size'
   23   'Default IP Time-to-live'
   24   'Path MTU Aging Timeout'
   25   'Path MTU Plateau Table'
   26   'MTU Option'
   27   'All subnets are local'
   29   'Perform Mask Discovery'
   30   'Mask Supplier Option'
   31   'Perform Router Discovery'
   32   'Router Solicitation Address'
   33   'Static Route Option'
   34   'Trailer Encapsulation'
   35   'ARP Cache Timeout'
   36   'Ethernet Encapsulation'
   37   'TCP Default Time-to-live'
   38   'Keepalive Interval'
   39   'Keepalive Garbage'
   40   'NIS Domain Name'
   41   'NIS Servers'
   42   'NTP Servers'
   43   'Vendor Specific Info'
   44   'WINS/NBNS Servers'
   45   'NetBIOS over TCP/IP NBDD'
   46   'WINS/NBT Node Type'
   47   'NetBIOS Scope ID'
   48   'X Window System Font'
   49   'X Window System Display'
   64   'NIS+ Domain Name'
```

```
65   'NIS+ Servers'
66   'Boot Server Host Name'
67   'Bootfile Name'
68   'Mobile IP Home Agents'
69   'Simple Mail Transport Protocol (SMTP) Servers'
70   'Post Office Protocol (POP3) Servers'
71   'Network News Transport Protocol (NNTP) Servers'
72   'World Wide Web (WWW) Servers'
73   'Finger Servers'
74   'Internet Relay Chat (IRC) Servers'
75   'StreetTalk Servers'
76   'StreetTalk Directory Assistance (STDA) Servers'
```

CONSTRUCTOR

```
  my $option = Infoblox::DHCP::MSOption->new(
     value        => $num | $string |"true"|"false", #Required
     name         => $string,                        #Required
     code         => $int,                           #Required
     type         => $string,                        #Required
     vendor_class => $string,                        #Optional / If not specified, it uses the default
Microsoft vendor class
     user_class   => $string,                        #Optional / If not specified, it uses the default
Microsoft user class
   );
```

MODULE METHODS

The following functions can be applied to a DHCP Option object.

Infoblox::Grid::MSServer::DHCP->ms_options( )

Use this function to specify DHCP options at the Microsoft server level. See Infoblox::Grid::MSServer::DHCP->ms_options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $opt1 = Infoblox::DHCP::MSOption->new(
                                         name         => 'Time Offset',
                                         code         => 2,
                                         type         => 'long',
                                         value        => 50,
                                         vendor_class => 'DHCP Standard Options',
                                         user_class   => 'Default User Class',
                                        );
# Configure a DHCP option for a Microsoft DHCP server
my $response = $MSServer_DHCP->options([$opt1]);
```

Infoblox::DHCP::Range->ms_options( )

Use this function to specify Microsoft DHCP options at the range level. See Infoblox::DHCP::Range->ms_options() for parameters and return values.

Example

```
my $opt1 = Infoblox::DHCP::MSOption->new(
                                         name         => 'LPR Servers',
                                         code         => 9,
                                         type         => 'array of ip address',
                                         value        => '10.4.1.0,10.4.1.1,10.4.1.2',
                                         vendor_class => 'DHCP Standard Options',
                                         user_class   => 'Default User Class',
                                        );
# Configure a Microsoft DHCP option for a range
my $response = $Range->ms_options([$opt1]);
```

Infoblox::DHCP::FixedAddr->options( )

Use this function to specify DHCP options at the fixed address level. See Infoblox::DHCP::FixedAddr->ms_options() for parameters and return values.

Example

```
my $opt1 = Infoblox::DHCP::MSOption->new(
                                         name         => 'Static Route Option',
```

```
                                        code          => 33,
                                        type          => 'array of ip address pair',
                                        value         => '10.5.1.0 10.5.1.1,10.5.1.2 10.5.1.3',
                                       );
# Configure a DHCP option for a network
my $response = $FixedAddr->ms_options([$opt1]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP option.

name( )

Use this method to set or retrieve the name of the DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the name of the Microsoft DHCP option in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $option->name();
#Modify name
$option->name("Static Route Option");
```

code( )

Use this method to set or retrieve the code value of the DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the code of the DHCP option. It must be an integer from 1 to 254.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get code
my $code = $option->code();
#Modify code
$option->code("12");
```

type( )

Use this method to set or retrieve the type value of the DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the type of the DHCP option. It must be one of the following values:

```
'binary'
'byte'
'array of byte'
'array of ip address pair'
'ip address'
'array of ip address'
'long integer'
'array of long integer'
'long'
'array of long'
'array of string'
'array of word'
'encapsulated'
```

```
'string'
'word'
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get type
my $type = $option->type();
#Modify type
$option->type("string");
```

value( )

Use this method to set or retrieve the custom value of the DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the data type of the DHCP option. If the type is set to 'array of...', the accepted type is a list of the specified types.

```
"binary"                          - Binary data.
"byte"                            - Specify a numeric value between 0 and 255.
"encapsulated"                    - Encapsulated data.
"ip address"                      - Specify an IPv4 address.
"ip address pair"                 - Specify a pair of IPv4 addresses separated by a space.
"long"                            - Specify a numeric value between 0 and 4294967295.
"long integer"                    - Specify a numeric value between 0 and 1.84467441   10^19
"string"                          - Specify either a value in ASCII text string or
                                    a list of hexadecimal characters (0-9, a-f, or A-F) without
spaces and separated by colons.
"word"                            - Specify a numeric value between 0 and 65535.
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get value
my $value = $option->value();
#Modify value
$option->value("abc.com");
```

user_class( )

Use this method to set an option space name as the user class value of a DHCP option or to retrieve an option space name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the defined option space name in string format. The default user class name is "Default User Class".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get user_class
my $name = $option->user_class();
#Modify user_class
$option->user_class("Default User Class");
```

vendor_class( )

Use this method to set an option space name as the vendor class value of a DHCP option or to retrieve an option space name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the defined option space name in string format. The default vendor space name is "DHCP Standard Options".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get vendor_class
my $name = $option->vendor_class();
#Modify vendor_class
$option->vendor_class("DHCP Standard Options");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify(),Infoblox::Grid::MSServer::DHCP, Infoblox::DHCP::Range, Infoblox::DHCP::FixedAddr

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::MSServer - Microsoft DHCP Server object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::FixedAddr->member( ), Infoblox::DHCP::Network->members( ), Infoblox::DHCP::NetworkTemplate->members( ), Infoblox::DHCP::RangeTemplate->member( ), Infoblox::DHCP::Range->member( )
- METHODS
  - address( )
  - name( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::MSServer - Microsoft (r) DHCP Server object.

DESCRIPTION

This object is used to reference a Microsoft DHCP server.

CONSTRUCTOR

```
my $member = Infoblox::DHCP::MSServer->new(
    address  => $address, #Required
    name     => $name,    #Optional
);
```

MODULE METHODS

This section describes all the functions that you can apply to a DHCP MSServer object.

Infoblox::DHCP::FixedAddr->member( ), Infoblox::DHCP::Network->members( ), Infoblox::DHCP::NetworkTemplate->members( ), Infoblox::DHCP::RangeTemplate->member( ), Infoblox::DHCP::Range->member( )

Use this function to specify a member for any of the above objects. See their respective documentation pages for parameters and return values.

Example

```
my $member1 = Infoblox::DHCP::MSServer->new(
    address => "10.0.0.1",
);

#Create the Network object with this member
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/255.0.0.0",
    members => [$member1],
);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a DHCP MSServer object.

address( )

Use this method to set or retrieve the address of the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The Microsoft DNS server address in IPV4 format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get address
my $address = $member->address();
#Modify address
$member->address("192.168.1.3");
```

name( )

Use this method to set or retrieve the name of the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The Microsoft DNS server name in FQDN format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $member->name();
#Modify name
$member->name("msserver.com");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP MSServer object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
      die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a grid member object

my $grid_member = Infoblox::Grid::Member->new(
    name          => "hostname.com",
    ipv4addr      => "3.0.0.2",
    mask          => "255.0.0.0",
    gateway       => "3.0.0.1",
    comment       => "Script Generated"
    );
unless ($grid_member) {
      die("Construct grid member object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Grid member object created successfully\n";

#Adding the member object to appliance through session
```

```
my $response = $session->add( $grid_member );

unless ($response) {
    die("Add grid member failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print"Grid member added successfully \n";

my $msserver = Infoblox::Grid::MSServer->new(
                                            address                  => '10.0.0.1',
                                            comment                  => 'basic member',
                                            disable                  => 'true',
                                            extensible_attributes     => { Site => 'Local'},
                                            login                    => 'user',
                                            password                 => 'password',
                                            managing_member           => 'hostname.com',
                                            read_only                => 'false',
                                            synchronization_interval => 5,
                                            logging_mode             => 'minimum',
                                           );

unless ($msserver) {
       die("Construct grid MS DNS server object failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Grid MS DNS server object created successfully\n";

#Adding the member object to appliance through session
$response = $session->add( $msserver );

unless ($response) {
    die("Add grid MS DNS server failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print"Grid MS DNS server added successfully \n";
```

#Create a DHCP MSServer object

```
my $member1 = Infoblox::DHCP::MSServer->new(
    address => "10.0.0.1",
);

unless($member1) {
     die("Construct member failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create the Network object with this member
my $network = Infoblox::DHCP::Network->new(
                    network => "10.0.0.0/255.0.0.0",
                    comment => "add network",
                    members => [ $member1 ]
);

unless($network) {
            die("Construct Network object failed: ",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
     or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP member object added to the network successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::Network, Infoblox::Session->add(), Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::MSSuperscope - Microsoft Server Superscope object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - name( )
  - network_view( )
  - ranges( )
  - range_high_water_mark( )
  - range_low_water_mark( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::MSSuperscope - Microsoft (r) Server Superscope object.

DESCRIPTION

A superscope is a feature of Microsoft DHCP servers. You can use a superscope to group and activate multiple ranges via a single object.

CONSTRUCTOR

```
 my $superscope = Infoblox::DHCP::Superscope->new(
                 name                          => $string,                        #Required
                 ranges                        => [ $dhcprange1,$dhcprange2,...],   #Required
                 comment                       => $string,                        #Optional
                 extensible_attributes          => { $string => $string | $num, $string => [ $string |
$num, ... ], ... },
                                                                                  #Optional / Default
is undefined
                 network_view                  => $NetworkView,                    #Optional / Default
is the default network view
                 disable                       => "true" | "false",                #Optional / Default
is "false"
            )
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a superscope object.

Infoblox::Session->add( )

Use this method to add a superscope object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
       #Construct an object
       my $superscope = Infoblox::DHCP::MSSuperscope->new(
                                                   name                 => 'test superscope',
                                                   comment              => 'test comment',
                                                   disable              => 'true',
                                                   network_view         => $view,
                                                   ranges               => [ $dhcp_range ],
                                                  );
```

```
# Submit for addition
my $response = $session->add( $superscope );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific superscope object:

name                   - Optional. Name value for the superscope.
comment                - Optional. Comment value for the superscope.
network_view           - Optional. Network view in which the superscope is located. By default, all
network views are searched.
extensible_attributes  - Optional. A hash reference containing extensible attributes.
```

Examples

```
my @retrieved_objs = $session->get(
                     object     => "Infoblox::DHCP::MSSuperscope",
                     name       => "test superscope" );

my @retrieved_objs = $session->get(
                     object       => "Infoblox::DHCP::MSSuperscope",
                     name         => "other superscope",
                     network_view => "external" );

my @retrieved_objs = $session->get(
                     object                => "Infoblox::DHCP::MSSuperscope",
                     extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a superscope object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment.
$superscope->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $superscope );
```

Infoblox::Session->remove( )

Use this method to remove a superscope object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the superscope object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::MSSuperscope",
    name         => "some superscope",
    newtork_view => "default" );
# Find the desired object from the retrieved list.
my $desired_superscope = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_superscope );
```

Infoblox::Session->search( )

Use this method to search for superscope objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a superscope object:

 name                   - Optional. Name value for the superscope (regular expression).
 comment                - Optional. Comment value for the superscope (regular expression).
 network_view           - Optional. Network view in which the superscope is located. By default, all
network views are searched.
 extensible_attributes  - Optional. A hash reference containing extensible attributes.
```

Examples

```
# search for superscope objects with comments ending with 'comment'
my @retrieved_objs = $session->search(
    object     => "Infoblox::DHCP::MSSuperscope",
    comment    => ".*comment" );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a superscope.

comment( )

Use this method to set or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $superscope->comment();
#Modify comment
$superscope->comment("Modifying the Superscope comment");
```

disable( )

Use this method to set or retrieve the disable flag of a superscope.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value for this field is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $superscope->disable();
#Modify disable
$superscope->disable("false");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes that are associated with a superscope object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $superscope->extensible_attributes();
#Modify extensible attributes
$superscope->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

name( )

Use this method to set or retrieve the name of the superscope.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the name of the superscope in string format. The string must not be longer than 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $superscope->name();
#Modify name
$superscope->name("superscope name");
```

network_view( )

Use this method to set or retrieve the network view of the superscope.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $superscope->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$superscope->network_view($nview);
```

ranges( )

Use this method to set or retrieve the DHCP ranges that are associated with this superscope.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::Range objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ranges
my $ref_options = $superscope->ranges();
#Modify ranges
$superscope->ranges([$range]); #$range is an Infoblox::DHCP::Range object
```

range_high_water_mark( )

Use this method to retrieve the percent above "Normal" when the DHCP utilization status becomes "High".

=over

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get high-water mark
my $range_high_water_mark = $superscope->range_high_water_mark();
```

range_low_water_mark( )

Use this method to retrieve the percent below "Normal" when the DHCP utilization status becomes "Low".

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get low-water mark
my $range_low_water_mark = $superscope->range_low_water_mark();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a superscope object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP superscope object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create supporting objects, $host_name is the grid member's fqdn

```
my $ms_member = Infoblox::Grid::MSServer->new(
                                          address                  => '10.0.0.1',
                                          comment                  => 'basic member',
                                          disable                  => 'true',
                                          extensible_attributes    => { Site => 'Somewhere'},
                                          login                    => 'user',
                                          password                 => 'password',
                                          managing_member          => $host_name,
                                          read_only                => 'false',
                                          synchronization_interval => 5,
                                          logging_mode             => 'minimum',
                                         );

unless($ms_member) {
       die("Construct MS Member object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
}
print "MS Member object created successfully\n";

unless($session->add($ms_member)) {
       die("Add MS Member object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "MS Member object added successfully\n";

my $ms_dhcp = Infoblox::DHCP::MSServer->new(address => '10.0.0.1');

unless($ms_dhcp) {
       die("Construct MS DHCP Member object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "MS DHCP Member object created successfully\n";

my $network1 = Infoblox::DHCP::Network->new(
                                          network   => "10.0.0.0/255.255.0.0",
                                          comment   => "add network",
                                          members   => [$ms_dhcp],
                                         );
unless($network1) {
       die("Construct DHCP Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Network object created successfully\n";

unless($session->add($network1)) {
       die("Add DHCP Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Network object added successfully\n";

my $dhcp_range1 = Infoblox::DHCP::Range->new(
                                           end_addr                => "10.0.0.10",
                                           network                 => "10.0.0.0/16",
                                           start_addr              => "10.0.0.1",
                                           disable                 => "true",
                                           member                  => $ms_dhcp,
                                          );

unless($dhcp_range1) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

unless($session->add($dhcp_range1)) {
       die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Range object added successfully\n";

my $network2 = Infoblox::DHCP::Network->new(
                                          network   => "10.1.0.0/255.255.0.0",
                                          comment   => "add network",
                                          members   => [$ms_dhcp],
                                         );
unless($network2) {
       die("Construct DHCP Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Network object created successfully\n";

unless($session->add($network2)) {
       die("Add DHCP Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Network object added successfully\n";

my $dhcp_range2 = Infoblox::DHCP::Range->new(
                                           end_addr                => "10.1.0.30",
                                           network                 => "10.1.0.0/16",
                                           start_addr              => "10.1.0.21",
```

```
                                                disable                 => "true",
                                                member                  => $ms_dhcp,
                                               );

unless($dhcp_range2) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

unless($session->add($dhcp_range2)) {
       die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Range object added successfully\n";

#Create a DHCP Superscope object

my $superscope = Infoblox::DHCP::MSSuperscope->new(
                                                    name                    => 'test superscope',
                                                    comment                 => 'test comment',
                                                    disable                 => 'true',
                                                    ranges                  => [ $dhcp_range1, $dhcp_range2],
                                                    );
unless($superscope) {
       die("Construct DHCP Superscope object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Superscope object created successfully\n";

unless($session->add($superscope)) {
       die("Add DHCP Superscope object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Superscope object added successfully\n";

#Get the DHCP Superscope object and change its ranges

#Get the DHCP Superscope object
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::MSSuperscope",
    name   => "test superscope"
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP Superscope object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP Superscope object found at least 1 matching entry\n";

#Apply the changes
$session->modify($object)
    or die("Modify DHCP Superscope object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Superscope object with new range assignments updated to Infoblox appliance successfully\n";

#Remove DHCP Superscope and support objects

$session->remove($object)
    or die("Remove DHCP Superscope object failed: ",
            $session->status_code() . ":" . $session->status_detail());

$session->remove($dhcp_range2)
    or die("Remove DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());

$session->remove($dhcp_range1)
    or die("Remove DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());

$session->remove($network2)
    or die("Remove DHCP Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());

$session->remove($network1)
    or die("Remove DHCP Network object failed: ",
```

```
            $session->status_code() . ":" . $session->status_detail());

$session->remove($ms_member)
    or die("Remove Grid MSServer object failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "All objects removed from the Infoblox appliance successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify(),Infoblox::DHCP::Range,Infoblox::DHCP::MSOption

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Network - DHCP Network object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - authority( )
  - auto_create_reversezone( )
  - bootfile( )
  - bootserver( )
  - comment( )
  - ddns_domainname( )
  - ddns_generate_hostname( )
  - ddns_server_always_updates( )
  - ddns_ttl( )
  - ddns_update_fixed_addresses( )
  - ddns_use_option81( )
  - deny_bootp( )
  - disable( )
  - email_list( )
  - enable_ddns( )
  - enable_dhcp_thresholds( )
  - enable_email_warnings( )
  - enable_snmp_warnings( )
  - extensible_attributes( )
  - high_water_mark( )
  - ignore_dhcp_option_list_request( )
  - lease_scavenge_time( )
  - low_water_mark( )
  - members( )
  - network( )
  - network_container( )
  - network_view( )
  - nextserver( )
  - next_available_ip( )
  - options( )
  - override_ddns_domainname( )
  - override_ddns_ttl( )
  - override_lease_scavenge_time( )
  - override_update_dns_on_lease_renewal( )
  - pxe_lease_time( )
  - recycle_leases( )
  - resize( )
  - template( )
  - zone_associations( )
  - update_dns_on_lease_renewal( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Network - DHCP Network object

DESCRIPTION

When DHCP services are configured on an appliance, a network must be defined that it serves. After a network is created, you can then create all the subnetworks individually or create a parent network that encompasses all the subnetworks.

CONSTRUCTOR

```
my $network = Infoblox::DHCP::Network->new(
```

```
                    network                         => $ipv4_subnet/prefix,                 #Required, when
creating a network
                    ipv4addr                        => $ipv4addr,                           #Use this parameter
instead of "network", when using a template with 'allow_any_netmask' set to false to create a network.
                    authority                       => "true" | "false"| undef,             #Optional / Default
is undefined
                    auto_create_reversezone         => "true" | "false"| undef,             #Optional / Default
is undefined
                    bootfile                        => $filename | undef,                   #Optional / Default
is undefined
                    bootserver                      => $ipv4addr | $fqdn | undef,           #Optional / Default
is undefined
                    comment                         => $string| undef,                      #Optional / Default
is undefined
                    ddns_domainname                 => $string| undef,                      #Optional / Default
is undefined
                    ddns_generate_hostname          => "true" | "false" | undef,            #Optional / Default
is undefined
                    ddns_server_always_updates      => "true" | "false",                    #Optional / Default
is "true"
                    ddns_ttl                        => $num,                                #Optional / Default
is "0"
                    ddns_use_option81               => "true" | "false" | undef,            #Optional / Default
is undefined
                    ddns_update_fixed_addresses     => "true" | "false" | undef,            #Optional / Default
is undefined
                    deny_bootp                      => "true" | "false" | undef,            #Optional / Default
is undefined
                    disable                         => "true" | "false",                    #Optional / Default
is "false"
                    email_list                      => [ "admin\@test.com",...] | undef,    #Optional / Default
is undefined
                    enable_ddns                     => "true" | "false" | undef,            #Optional / Default
is undefined
                    enable_dhcp_thresholds          => "true" | "false",                    #Optional / Default
is "false"
                    enable_email_warnings           => "true" | "false",                    #Optional / Default
is "false"
                    enable_snmp_warnings            => "true" | "false",                    #Optional / Default
is "false"
                    extensible_attributes           => { $string => $string | $num, $string => [ $string |
$num, ... ], ... } #Optional / Default is undefined
                    high_water_mark                 => $string,                             #Optional / Default
is "95"
                    ignore_dhcp_option_list_request => "true" | "false" | undef,            #Optional / Default
is undefined
                    lease_scavenge_time             => $number,                             #Optional / Default
is "-1"
                    low_water_mark                  => $string,                             #Optional / Default
is "0"
                    members                         => [ $Member1,$Member2,...],            #Optional / Default
is empty
                    network_container               => $ipv4_subnet/prefix,                 #Optional / Default
is "/"
                    network_view                    => $NetworkView,                        #Optional / Default
is "default" network view
                    nextserver                      => $ipv4addr | $fqdn | undef,           #Optional / Default
is undefined
                    options                         => [$Option1,$Option2,...],             #Optional / Default
is empty list
                    override_ddns_domainname        => "true" | "false",                    #Optional / Default
value is "false"
                    override_ddns_ttl               => "true" | "false",                    #Optional / Default
value is "false"
                    override_lease_scavenge_time    => "true" | "false",                    #Optional / Default
value is "false"
                    override_update_dns_on_lease_renewal  => "true" | "false",              #Optional / Default
value is "false"
                    pxe_lease_time                  => $num | undef,                        #Optional / Default
is undefined
                    recycle_leases                  => "true" | "false" | undef,            #Optional / Default
is undefined
                    template                        => $string | undef,                     #Optional / Default
is undefined
                    update_dns_on_lease_renewal     => "true" | "false",                    #Optional / Default
value is "false"
                    zone_associations               => [$zone1, $zone2, ... ] | undef,      #Optional / Default
is an empty list
                );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that can apply to a DHCP Network object.

Infoblox::Session->add( )

Use this method to add a network object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values. Notice the syntax difference between simple adding a network and adding a network using a template.

Example

```
#Adding a network.
#Construct an object
my $network = Infoblox::DHCP::Network->new(
      network => "20.0.0.0/8",
);

# Submit for adding a network
my $response = $session->add( $network );

#Adding a network using template
#Construct an object
my $network = Infoblox::DHCP::Network->new(
      ipv4addr => '20.0.0.0',
      template => "mytemplate",
);

# Submit for adding a network
my $response = $session->add( $network );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching network objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP Network object:

  network               - Optional. The IP address and netmask of a network in CIDR format.
  network_container     - Optional. The network container, in CIDR format, to which this network
belongs.
  network_view          - Optional. The network view in which the network is located. By default, all
network views are searched (except when contain_address is specified -- see below).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
  comment               - Optional. A comment in string format (regular expression).
  contains_address      - Optional. An IPv4 address in string format. If specified, all other search
attributes are ignored, except for network_view. When a value for this attribute is provided,
network_view defaults to the system-defined default network view, unless specified. The returned
network is the smallest network enclosing this IPv4 address.
  member                - Optional. The member or Microsoft server serving DHCP for the network. The
valid value is an Infoblox::DHCP::Member or Infoblox::DHCP::MSServer object.
```

Examples

```
my  @result_array = $session->get(
     object        => "Infoblox::DHCP::Network" ,
     network       => "20.0.0.0/8",
     network_view => "default"
);

my  @result_array = $session->get(
     object                => "Infoblox::DHCP::Network" ,
     extensible_attributes => { 'Site' => 'Santa Clara' }
);

my  @result_array = $session->get(
     object  => "Infoblox::DHCP::Network" ,
     comment => "my-comment",
);

my  @result_array = $session->get(
     object  => "Infoblox::DHCP::Network" ,
     network_container => "10.0.0.0/16",
);

# Returns the smallest network enclosing 10.0.0.1 in the
# system-defined default network view.
my  @result_array = $session->get(
     object  => "Infoblox::DHCP::Network" ,
     contains_address => "10.0.0.1",
);
```

```
# Returns the smallest network enclosing 10.0.0.1 in the
# specified network view ($network_view_object).
my  @result_array = $session->get(
     object  => "Infoblox::DHCP::Network" ,
     contains_address => "10.0.0.1",
     network_view => $network_view_object,
);
```

Infoblox::Session->modify( )

Use this method to modify a network object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify enable_ddns method.
$object->enable_ddns("true");
# Submit modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my  @result_array = $session->get(

     object        => "Infoblox::DHCP::Network" ,
     network       => "20.0.0.0/8",
     network_view  => "default"
 );

# find the desired object from the retrieved list.
my $desired_network = $result_array[0];

# Submit for removal
my $response = $session->remove( $desired_network );
```

Infoblox::Session->search( )

Use this method to search for a DHCP Network object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP Network object:

  network                - Optional. The IP address and netmask of a network in CIDR format.
  network_container      - Optional. The network container, in CIDR format, to which this network
belongs.
  network_view           - Optional. The network view in which the network is located. By default, all
network views are searched.
  extensible_attributes - Optional. A hash reference containing extensible attributes.
  comment                - Optional. A comment in string format (regular expression).
  contains_address       - Optional. An IPv4 address in string format. If specified, all other search
attributes are ignored, except for network_view. When a value for this attribute is provided,
network_view defaults to the system-defined default network view, unless specified. The returned
network is the smallest network enclosing this IPv4 address.
  member                 - Optional. The member or Microsoft server serving DHCP for the network. The
valid value is an Infoblox::DHCP::Member or Infoblox::DHCP::MSServer object.
```

Examples

```
# search for all DHCP Network objects
    my @retrieved_objs = $session->search(
      object       => "Infoblox::DHCP::Network",
      network      => '20\.0\.0\..*',
      network_view => "default"
    );

# search network by comment
    my @retrieved_objs = $session->search(
    object    => "Infoblox::DHCP::Network",
    comment   => "my-comment",
     );
```

```
# search network by network container
 my @result_array = $session->search(
      object  => "Infoblox::DHCP::Network" ,
      network_container => "10.0.0.0/16",
 );

# search for all DHCP Network objects defining "Santa Clara" for "Site" extensible attribute
    my @retrieved_objs = $session->search(
      object                => "Infoblox::DHCP::Network",
      extensible_attributes => { 'Site' => 'Santa Clara' }
    );

# Returns the smallest network enclosing 10.0.0.1 in the
# system-defined default network view.
    my @result_array = $session->search(
       object  => "Infoblox::DHCP::Network" ,
       contains_address => "10.0.0.1",
    );

# Returns the smallest network enclosing 10.0.0.1 in the
# specified network view ($network_view_object).
    my @result_array = $session->search(
       object  => "Infoblox::DHCP::Network" ,
       contains_address => "10.0.0.1",
       network_view => $network_view_object,
    );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP Network object

authority( )

Use this method to set or retrieve the authority of a DHCP network object. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the authority or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute inherit the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get authority
my $authority = $network->authority()
# Modify authority
$network->authority("true");
```

auto_create_reversezone( )

Use this method to set the auto_create_reversezone flag of a DHCP network object.

Include the specified parameter to set the attribute value. This is a write-only attribute and cannot be retrieved after it is set.

Parameter

Specify "true" to set the auto_create_reversezone flag or "false" to deactivate/unset it. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
# Modify auto_create_reversezone
$network_container->auto_create_reversezone("true");
```

bootfile( )

Use this method to set or retrieve a bootfile attribute of a DHCP network object. You can configure the DHCP server to support clients

that use boot file name option in their DHCPREQUEST messages. This setting overrides the upper-level settings. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::Network->bootserver() and Infoblox::DHCP::Network->nextserver().

Parameter

The name, in string format, of the file that the client must download. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get bootfile
my $bootfile = $network->bootfile();
# Modify bootfile
$network->bootfile("bootfile1");
#Un-override bootfile
$network->bootfile(undef);
```

bootserver( )

Use this method to set or retrieve a bootserver attribute of a DHCP network object. You can specify the name and/or IP address of the boot server that host needs to boot. This setting overrides the upper-level settings. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::Network->bootfile() and Infoblox::DHCP::Network->nextserver().

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get bootserver
my $bootserver = $network->bootserver();
# Modify bootserver
$network->bootserver("abc.domain.com");
#Un-override bootserver
$network->bootserver(undef);
```

comment( )

Use this method to set or retrieve a descriptive comment of a DHCP network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $network->comment();
# Modify comment
$network->comment("add a network");
```

ddns_domainname( )

Use this method to set or retrieve the ddns_domainname value of a DHCP Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_dddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

The dynamic DNS domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_domainname
my $ddns_domainname = $network->ddns_domainname();
#Modify ddns_domainname
$network->ddns_domainname("test_domain.com");
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname flag of a DHCP network object. Enable this method to allow only the DHCP server for generating hostname. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to generate hostname or "false" to deactivate/unset it.The default value for this field is undefined, which indicates that this attribute inherit the member level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_generate_hostname
my $ddns_generate_hostname = $network->ddns_generate_hostname()
# Modify ddns_generate_hostname
$network->ddns_generate_hostname("true");
#Un-override
$network->ddns_generate_hostname(undef);
```

ddns_server_always_updates( )

Use this method to set or retrieve the ddns_server_always_updates flag of a DHCP network object. Enable this method to allow only the DHCP server to update DNS,regardless of the requests from the DHCP clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the flag that enables the DHCP server to always updates DNS or "false" to deactivate/unset it.The default value for this field is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_server_always_updates
my $ddns_server_always_updates = $network->ddns_server_always_updates()
# Modify ddns_server_always_updates
$network->ddns_server_always_updates("true");
```

ddns_ttl( )

Use this method to set or retrieve the DNS update Time to Live (TTL) value of a DHCP network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_ttl method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_ttl attribute to "false".

Note that when ddns_ttl is set to a defined value and override_dddns_ttl is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_ttl(12); $object->override_ddns_ttl("false"); will set override_ddns_ttl to "false", and the sequence $object->override_ddns_ttl("false"); $object->ddns_ttl(12); will result in override_ddns_ttl="true".

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the update is cached. Zero indicates that the update is not cached. The default value is 0.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get DNS Update ttl
my $dns_ttl = $network->ddns_ttl();
#Modify DNS Update ttl
$network->ddns_ttl(1200);
```

ddns_update_fixed_addresses( )

Use this method to set or retrieve the ddns_update_fixed_addresses flag of a DHCP network object. By default, the DHCP server does not update DNS when it allocates a fixed address to a client. You can configure the DHCP server to update the A and PTR records of a client with a fixed address. When this feature is enabled and the DHCP server adds A and PTR records for a fixed address, the DHCP server never discards the records. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the appliance to update DNS when it assigns a fixed address to a client or "false" to disable it. The default value for this field is undefined, which indicates that this attribute inherit the member level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_update_fixed_addresses
my $ddns_update_fixed_addresses= $network->ddns_update_fixed_addresses()
# Modify ddns_update_fixed_addresses
$network->ddns_update_fixed_addresses("true");
#Un-override
$network->ddns_update_fixed_addresses(undef);
```

ddns_use_option81( )

Use this method to set or retrieve the ddns_use_option81 flag of a DHCP network object. Enable this method to define the settings for option 81 at the network level. This setting overrides the member level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to Configure Support for Option 81 or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute inherit the member level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_use_option81
my $ddns_use_option81 = $network->ddns_use_option81()
# Modify ddns_use_option81
$network->ddns_use_option81("true");
# un-override ddns_use_option81
$network->ddns_use_option81(undef);
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp flag of a DHCP Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_bootp flag or "false" to deactivate/unset it. The default value is undefined, which indicates that this attribute inherit the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $network->deny_bootp();
#Modify deny_bootp
$network->deny_bootp("true");
```

disable( )

Use this method to set or retrieve the disable flag of a DHCP network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get disable
my $disable = $network->disable()
# Modify disable
$network->disable("true");
```

email_list( )

Use this method to set or retrieve the e-mail lists to which the appliance sends threshold e-mail messages.

If you enter e-mail addresses in the email_list, then the appliance sends DHCP threshold alarms to the e-mail addresses.

Parameter

The valid value is an array reference that contains string objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get email_list
my @email_list = $network->email_list();
#Modify email_list
$network->email_list( [ "admin\@test.com" ] );
```

enable_ddns( )

Use this method to set or retrieve the dynamic DNS updates flag of a DHCP network object. The DHCP server can send DDNS updates to DNS servers in the same grid and to external DNS servers. This setting overrides the member level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the dynamic DNS updates flag or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute inherit the member level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_ddns
my $enable_ddns = $network->enable_ddns()
# Modify enable_ddns
$network->enable_ddns("true");
# Un-override enable_ddns
$network->enable_ddns(undef);
```

enable_dhcp_thresholds( )

Use this method to set or retrieve DHCP thresholds at the network level. These thresholds are the high and low watermarks. If a DHCP range usage goes above or below these percentage values, the Infoblox appliance generates a syslog message and sends an SNMP notification.

Enable this flag to set DHCP thresholds values.

Parameter

Specify "true" to set the enable_dhcp_thresholds flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcp_thresholds
my $enable_dhcp_thresholds = $network->enable_dhcp_thresholds();
#Modify enable_dhcp_thresholds
$network->enable_dhcp_thresholds("false");
```

enable_email_warnings( )

Use this method to send DHCP threshold warnings through e-mail.

Enable this flag if you want the appliance to send DHCP threshold warnings to the "email_list" provided.

Parameter

Specify "true" to set the enable_email_warnings flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_email_warnings
my $enable_email_warnings = $network->enable_email_warnings();
#Modify enable_email_warnings
```

```
$network->enable_email_warnings("false");
```

enable_snmp_warnings( )

Use this method to send DHCP threshold warnings through SNMP.

If the flag is enabled, then the appliance sends DHCP threshold warnings through SNMP messages.

Parameter

Specify "true" to set the enable_snmp_warnings flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_snmp_warnings
my $enable_snmp_warnings = $network->enable_snmp_warnings();
#Modify enable_snmp_warnings
$network->enable_snmp_warnings("false");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP network object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $network->extensible_attributes();
#Modify extensible attributes
$network->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

high_water_mark( )

Use this method to set the percentage value for DHCP range usage above which the Infoblox appliance generates a syslog message and sends a warning.

When the percent of allocated addresses in a DHCP range exceeds the high watermark, the appliance generates a syslog message and sends a warning. Default value for this option is 95.

Parameter

The valid value is in sting format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get high_water_mark
my $high_water_mark = $network->high_water_mark();
#Modify high_water_mark
$network->high_water_mark("80");
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the ignore_dhcp_option_list_request flag of a DHCP Network object. If this flag is set to false, the appliance returns all the defined DHCP options to the client. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate/unset it. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $network->ignore_dhcp_option_list_request();
# Modify ignore_dhcp_option_list_request
$network->ignore_dhcp_option_list_request("true");
#Do not override ignore_dhcp_option_list_request
$network->ignore_dhcp_option_list_request(undef);
```

lease_scavenge_time( )

Use this method to set or retrieve the lease_scavenge_time value of a DHCP Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_lease_scavenge_time method to "true". Setting the parameter to undefined causes the appliance to use the default from the upper level and automatically resets the override_lease_scavenge_time attribute to "false".

Note that when lease_scavenge_time is set to a defined value and override_lease_scavenge_time is set to "false", the last operation takes precedence. Thus the sequence $object->lease_scavenge_time(604800); $object->override_lease_scavenge_time("false"); will set override_lease_scavenge_time to "false", and the sequence $object->override_lease_scavenge_time("false"); $object->lease_scavenge_time(604800); will result in override_lease_scavenge_time="true".

Parameter

An integer that specifies the period of time, in seconds, that free and backup leases remain in the database before they are automatically deleted. To disable lease scavenging, set the parameter to -1. The minimum positive value must be greater than 86400 seconds (1 day). The default value of this parameter is -1, which means that lease scavenging is disabled.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lease_scavenge_time
my $lease_scavenge = $network->lease_scavenge_time();
#Modify lease_scavenge_time
$network->lease_scavenge_time(30 * 24 * 60 * 60);
```

low_water_mark( )

Use this method to set the percentage value for DHCP range usage below which the Infoblox appliance generates a syslog message and sends a warning.

When the percent of allocated addresses in a DHCP range goes below the low watermark, the appliance generates a syslog message and sends a warning. Default value for this option is 0.

Parameter

The valid value is in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get low_water_mark
my $low_water_mark = $network->low_water_mark();
#Modify low_water_mark
$network->low_water_mark("10");
```

members( )

Use this method to set or retrieve the member or Microsoft (r) server that serves DHCP for a network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Member or Infoblox::DHCP::MSServer objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $member1 = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
);
#Create the Network object with this member
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    members => [$member1],
);
```

network( )

Use this method to retrieve the network attribute of a DHCP network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network
my $network = $network->network();
```

network_container( )

Use this method to set or retrieve the network container of a DHCP network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the IP address and netmask of a network in CIDR format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network container
```

```
my $network_container = $network->network_container();
#Modify network container
$network->network_container("50.0.0.0/8");
```

network_view( )

Use this method to set or retrieve the network view of the DHCP network.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the network is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $network->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$network->network_view($nview);
```

nextserver( )

Use this method to set or retrieve the nextserver attribute of a DHCP network object. You can specify the name and/or IP address of the next server that the host needs to boot. This setting overrides the upper-level settings. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::Network->bootfile() and Infoblox::DHCP::Network->bootserver().

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get nextserver
my $nextserver = $network->nextserver();
# Modify nextserver
$network->nextserver("blue.domain.com");
#Un-override nextserver
$network->nextserver(undef);
```

next_available_ip( )

Use this method to retrieve the next available IP address in the network that was retrieved. Note that to use this method, the object MUST be retrieved through a get or search call, or it must have been created with an add call.

Parameter

This method does not accept input parameters.

Returns

This method returns the next available IP address in the network in string format.

Within a DHCP Network object, the next available IP address is the first address in the network that matches all of the following rules:

```
* There is no A, PTR, fixed address or host address record matching this IP address
* There is no bulk host record with this IP address
```

```
* There is no unmanaged record matching this IP address
* The IP address is not the network or broadcast address of this network
* The IP address is not in any DHCP range within this network (including any exclusion range)
```

Example

```
# Retrieve a particular network
my @retrieved_objs = $session->get(
                        object   => "Infoblox::DHCP::Network",
                        network  => "10.0.0.0/8"
);

$dhcp_network = $retrieved_objs[0];

# Print the next available IP address in the network
print $dhcp_network->next_available_ip() . "\n";
```

options( )

Use this method to set or retrieve the options value of a member DHCP server. DHCP options provide network configuration settings and various services available on the network for DHCP.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $network->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name => "lease-time",
    value => 50
);

$network->options([$option1]);
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP Network is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$network->override_ddns_domainname( );
#Modifying override_ddns_domainname
$network->override_ddns_domainname("true");
```

override_ddns_ttl( )

The override_ddns_ttl attribute controls whether the ddns_ttl value in the DHCP Network is used, instead of the grid default.

The override_ddns_ttl attribute can be specified explicitly. It is also set implicitly when ddns_ttl is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_ttl. Set the parameter to "false" to inherit the grid-level setting for ddns_ttl.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_ttl
my $override_ddns_ttl=$network->override_ddns_ttl( );
#Modifying override_ddns_ttl
$network->override_ddns_ttl("true");
```

override_lease_scavenge_time( )

The override_lease_scavenge_time attribute controls whether the lease_scavenge_time value in the DHCP network is used, instead of the upper-level default.

The override_lease_scavenge_time attribute can be specified explicitly. It is also set implicitly when lease_scavenge_time is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for lease_scavenge_time. Set the parameter to "false" to inherit the setting for lease_scavenge_time from the upper level.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_lease_scavenge_time
my $override_lease_scavenge_time=$network->override_lease_scavenge_time( );
#Modifying override_lease_scavenge_time
$network->override_lease_scavenge_time("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the DHCP Network is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$network->override_update_dns_on_lease_renewal( );
#Modifying override_update_dns_on_lease_renewal
$network->override_update_dns_on_lease_renewal("true");
```

pxe_lease_time( )

Use this method to set or retrieve the PXE lease time value of a DHCP Network object. Some hosts use PXE (Preboot Execution Environment) to boot remotely from a server. To better manage your IP resources, set a different lease time for PXE boot requests. You can configure the DHCP server to allocate an IP address with a shorter lease time to hosts that send PXE boot requests, so IP addresses are not leased longer than necessary. This setting overrides the upper-level settings. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the update is cached. Zero indicates that the update is not cached. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get PXE Lease Time
my $pxe_lease_time = $network->pxe_lease_time();
#Modify PXE Lease Time
$network->pxe_lease_time(1100);
#Un-override PXE Lease Time
$network->pxe_lease_time(undef);
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag at the network level.

If the flag is enabled, the leases are kept in the recycle bin until one week after expiration. When the flag is disabled, the leases are irrecoverably deleted.

Parameter

Specify "true" to enable the recycle_leases flag or "false" to disable it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $network->recycle_leases();
#Modify recycle_leases
$network->recycle_leases("false");
```

resize( )

Use this method to resize a network. Note that to use this method, the object MUST be retrieved through a get or search call, or it must have been created with an add call.

Parameter

prefix - Required. The bitmask prefix length after resizing.

auto_create_reversezone - Optional. Boolean flag indicating whether or not to automatically create the reverse zone.

Returns

The method returns true when the operation succeeds, and returns false when the operation fails. When an operation fails, the error code and error message are reported in the Session object's `status_code()` and `status_detail()` attributes.

Example

```
#Get Network through the session
```

```
my @retrieved_objs = $session->get(
          object =>"Infoblox::DHCP::Network" ,
          network => "20.0.0.0/8"
        );
my $network = $retrieved_objs[0];

# Expand a /24 network to a /16 network, inserting
# a corresponding reverse zone in the process.
my $response = $network->resize(
      prefix        => "16"
      auto_create_reversezone => "true"
) or die("Resize Network failed: ",
         $session->status_code() . ":" . $session->status_detail());
```

template( )

Use this method to set or retrieve the network template name of a DHCP Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid name in sting format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get template
my $template = $network->template();
#Modify template
$network->template("Template_Name");
```

zone_associations( )

Use this method to set or retrieve the associated zone information.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Zone object(s). The default value is an empty list. If the network has a parent, it is possible to use the zone associations of the parent by setting the parameter to undef.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Getting zone associations
my $zone_associations = $network->zone_associations();
my $zone1 = Infoblox::DNS::Zone->new(
      name     => "test1.com",
);
my $zone2 = Infoblox::DNS::Zone->new(
      name     => "test2.com",
   );
#Modifying zone_associations
$network->zone_associations([$zone1, $zone2]);
#Clear zone_assocations
$network->zone_associations([]);
##Remove the override for zone_associations
$network->zone_associations(undef)
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$network->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$network->update_dns_on_lease_renewal("true");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP Network insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP Network object

```
my $memberdhcp = Infoblox::DHCP::Member->new(
            name     => "infoblox.localdomain",
            ipv4addr => "192.168.1.2"
           );

my  $network = Infoblox::DHCP::Network->new(
            network => "10.0.0.0/8",
            comment => "add network",
            members => [$memberdhcp],
             );

my $response = $session->add($network)
       or die("Add Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network added successfully\n";

#Search for a specific Network

#Search the network
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::Network",
    network => '10\.0\.0\..*'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die
```

```
        ("Search Network failed: ",
         $session->status_code() . ":" . $session->status_detail());
}
print "Search Network found at least 1 matching entry\n";
```

#Get and modify a Network attribute

```
#Get Network through the session
my @retrieved_objs = $session->get(
           object =>"Infoblox::DHCP::Network" ,
           network => "10.0.0.0"
        );

my $object = $retrieved_objs[0];

unless ($object) {
    die("Get Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network found at least 1 matching entry\n";
```

#Modify attributes of the specified Network attribute

```
$object->enable_ddns("true"); #Apply the changes

#Un-override bootfile
$object->bootfile(undef);

#Un-override bootserver
$object->bootserver(undef);

#Un-override nextserver
$object->nextserver(undef);

$session->modify($object)
    or die("Modify Network attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "Network object modified successfully \n";
```

#Resize a network

```
#Get Network through the session
my @retrieved_objs = $session->get(
           object =>"Infoblox::DHCP::Network" ,
           network => "10.0.0.0/8"
        );
my $object = $retrieved_objs[0];

#Resize
my $response = $object->resize(
      prefix        => "16"
) or die("Resize Network failed: ",
         $session->status_code() . ":" . $session->status_detail());
```

#Remove a Network Object

```
 #Get Network object through the session
 my @retrieved_objs = $session->get(
           object  => "Infoblox::DHCP::Network" ,
           network => "10.0.0.0"
       );

   my $object = $retrieved_objs[0];

   unless ($object) {
    die("Get Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
    }

print "Get Network object found at least 1 matching entry\n";
```

#Submit the object for removal

```
$session->remove($object)
    or die("Remove Network failed: ",
        $session->status_code() . ":" . $session->status_detail());
```

```
print "Network removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::View, Infoblox::DHCP::Member, Infoblox::DHCP::Option, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::NetworkContainer - DHCP Network Container object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - auto_create_reversezone( )
  - comment( )
  - extensible_attributes( )
  - network( )
  - remove_subnets( )
  - network_container( )
  - network_view( )
  - resize( )
  - zone_associations( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::NetworkContainer - DHCP Network Container object

DESCRIPTION

A network can contain child networks. The network that contains the child network is called a network container.

CONSTRUCTOR

```
  my $network = Infoblox::DHCP::NetworkContainer->new(
                network                        => $ipv4_subnet/prefix,              #Required, when
creating a network container
                auto_create_reversezone         => "true" | "false",                #Optional / Default
is "false"
                comment                        => $string| undef,                   #Optional / Default
is undefined
                remove_subnets                 => "true" | "false",                 #Optional / Default
is undefined
                network_container               => $ipv4_subnet/prefix,              #Optional / Default
is "/"
                network_view                   => $NetworkView,                     #Optional / Default
is "default" network view
                zone_associations               => [$zone1, $zone2, ... ] | undef,    #Optional / Default
is an empty list
                extensible_attributes           => { $string => $string | $num, $string => [ $string |
$num, ... ], ... } #Optional / Default is undefined
              );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a DHCP Network Container object.

Infoblox::Session->add( )

Use this method to add a DHCP Network Container object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
       #Adding a network container.
       #Construct an object
       my $network_container = Infoblox::DHCP::NetworkContainer->new(
              network => "20.0.0.0/8",
```

```
);

# Submit for adding a network container
my $response = $session->add( $network_container );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching DHCP Network Container objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP Network Container object:

 network               - Optional. The IP address and netmask of a network in CIDR format.
 network_view          - Optional. The network view in which the network is located. By default, all
network views are searched.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 comment               - Optional. A comment in string format.
```

Examples

```
my  @result_array = $session->get(
     object        => "Infoblox::DHCP::NetworkContainer",
     network       => "20.0.0.0/8",
     network_view  => "default"
);

my  @result_array = $session->get(
     object                => "Infoblox::DHCP::NetworkContainer",
     extensible_attributes => { 'Site' => 'Santa Clara' }
);

my  @result_array = $session->get(
     object  => "Infoblox::DHCP::NetworkContainer",
     comment => "my-comment",
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP Network Container object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify comment.
$network_container->comment("modified comment");
# Submit modification
my $response = $session->modify( $network_container );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my  @result_array = $session->get(

      object        => "Infoblox::DHCP::NetworkContainer" ,
      network       => "20.0.0.0/8",
      network_view  => "default"
 );

# Find the desired object in the retrieved list.
my $network_container = $result_array[0];

# Submit for removal
my $response = $session->remove( $network_container );
```

Infoblox::Session->search( )

Use this method to search for a DHCP Network Container object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP Network Container object:

  network               - Optional. The IP address and netmask of a network in CIDR format (regular
expression).
  network_view          - Optional. The network view in which the network is located. By default, all
network views are searched.
  extensible_attributes - Optional. A hash reference containing extensible attributes.
  comment               - Optional. A comment in string format (regular expression).
```

Examples

```
# search for all DHCP Network Container objects
    my @retrieved_objs = $session->search(
      object       => "Infoblox::DHCP::NetworkContainer",
      network      => '20\.0\.0\..*',
      network_view => "default"
    );

# search network by comment
    my @retrieved_objs = $session->search(
      object    => "Infoblox::DHCP::NetworkContainer",
      comment   => "my-comment",
    );

# search for all DHCP Network Container objects defining "Santa Clara" for "Site" extensible
attribute
    my @retrieved_objs = $session->search(
      object                => "Infoblox::DHCP::NetworkContainer",
      extensible_attributes => { 'Site' => 'Santa Clara' }
    );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP Network Container object.

auto_create_reversezone( )

Use this method to set the auto_create_reversezone flag of a DHCP Network Container object.

Include the specified parameter to set the attribute value. This is a write-only attribute and cannot be retrieved after it is set.

Parameter

Specify "true" to set the auto_create_reversezone flag or "false" to deactivate/unset it. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
# Modify auto_create_reversezone
$network_container->auto_create_reversezone("true");
```

comment( )

Use this method to set or retrieve a descriptive comment for a DHCP Network Container object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network container in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $network_container->comment();
# Modify comment
$network_container->comment("add a network");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP Network Container object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $network_container->extensible_attributes();
#Modify extensible attributes
$network_container->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter',
'Tom' ] });
```

network( )

Use this method to retrieve the network attribute of a DHCP Network Container object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the address and netmask of the network in CIDR format (example: 10.0.0.0/8).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network
my $network = $network_container->network();
```

remove_subnets( )

Use this method to set or retrieve the remove_subnets flag of a DHCP Network Container object.

You can remove the network, but keep all subnets under it if this flag is set to false. This flag is only intended as an option when removing DHCP Network Container objects. It is not retrieved or used for any other operation.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "false" to keep the subnets or "true" to remove them. The default value is undefined, which is to remove all subnets.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Set remove_subnets before submitting for removal
$network_container->remove_subnets("false");
#Remove the container keeping the subnets
$session->remove($network_container);
```

network_container( )

Use this method to set or retrieve the network container of a DHCP Network Container object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the IP address and netmask of a network in CIDR format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network container
my $network_container = $network->network_container();
#Modify network container
$network->network_container("50.0.0.0/8");
```

network_view( )

Use this method to set or retrieve the network view of the DHCP network container.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the network is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $network_container->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$network_container->network_view($nview);
```

resize( )

Use this method to resize a network container. Note that to use this method, the object MUST be retrieved through a get or search call, or it must have been created with an add call.

Parameter

prefix - Required. The bitmask prefix length after resizing.

auto_create_reversezone - Optional. Boolean flag indicating whether or not to automatically create the reverse zone.

Returns

The method returns true when the operation succeeds, and returns false when the operation fails. When an operation fails, the error code and error message are reported in the Session object's `status_code()` and `status_detail()` attributes.

Example

```
#Get Network Container through the session
my @retrieved_objs = $session->get(
           object =>"Infoblox::DHCP::NetworkContainer" ,
           network => "20.0.0.0/8"
        );
my $network_container = $retrieved_objs[0];

# Expand a /24 network container to a /16 network container, inserting
# a corresponding reverese zone in the process.
my $response = $network_container->resize(
      prefix        => "16"
      auto_create_reversezone => "true"
) or die("Resize Network Container failed: ",
```

```
$session->status_code() . ":" . $session->status_detail());
```

zone_associations( )

Use this method to set or retrieve the associated zone information.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Zone object(s). The default value is an empty list. If the network container has a parent, it is possible to use the zone associations of the parent by setting the parameter to undef.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Getting zone associations
my $zone_associations = $network_container->zone_associations();
my $zone1 = Infoblox::DNS::Zone->new(
       name     => "test1.com",
);
my $zone2 = Infoblox::DNS::Zone->new(
       name     => "test2.com",
   );
#Modifying zone_associations
$network_container->zone_associations([$zone1, $zone2]);
#Clear zone_assocations
$network_container->zone_associations([]);
##Remove the override for zone_associations
$network_container->zone_associations(undef)
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP Network Container object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP Network Container object

```
my $network_container = Infoblox::DHCP::NetworkContainer->new(
            network => "10.0.0.0/8",
            comment => "add network container",
             );

my $response = $session->add($network_container)
       or die("Add Network Container failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network Container added successfully\n";

#Search for a specific Network Container

#Search the network container
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::NetworkContainer",
    network => '10\.0\.0\..*'
);
```

```
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search Network Container failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search Network Container found at least 1 matching entry\n";
```

#Get and modify a Network Container attribute

```
#Get Network Container through the session
my @retrieved_objs = $session->get(
           object =>"Infoblox::DHCP::NetworkContainer" ,
           network => "10.0.0.0/8"
        );
```

my $object = $retrieved_objs[0];

```
unless ($object) {
    die("Get Network Container object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network Container found at least 1 matching entry\n";
```

#Modify one of the attributes of the specified Network Container object

```
$object->comment("modified comment"); #Apply the changes

$session->modify($object)
    or die("Modify Network Container attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "Network Container object modified successfully \n";
```

#Resize a network container

```
#Get Network through the session
my @retrieved_objs = $session->get(
    object =>"Infoblox::DHCP::NetworkContainer" ,
    network => "10.0.0.0/8"
);
my $object = $retrieved_objs[0];

#Resize
my $response = $object->resize(
    prefix => "16"
) or die("Resize Network Container failed: ",
    $session->status_code() . ":" . $session->status_detail());
```

#Remove a Network Container Object

```
 #Get Network Container object through the session
 my @retrieved_objs = $session->get(
           object  => "Infoblox::DHCP::NetworkContainer" ,
           network => "10.0.0.0"
       );

   my $object = $retrieved_objs[0];

   unless ($object) {
    die("Get Network Container object failed: ",
        $session->status_code() . ":" . $session->status_detail());
    }

print "Get Network Container object found at least 1 matching entry\n";
```

#Submit the object for removal

```
$session->remove($object)
    or die("Remove Network Container failed: ",
        $session->status_code() . ":" . $session->status_detail());

print "Network Container removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::View, Infoblox::DHCP::Network, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::NetworkTemplate - DHCP Network Template object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
- METHODS
  - allow_any_netmask( )
  - authority( )
  - auto_create_reversezone( )
  - bootfile( )
  - bootserver( )
  - comment( )
  - ddns_domainname( )
  - ddns_generate_hostname( )
  - ddns_update_fixed_addresses( )
  - ddns_server_always_updates( )
  - ddns_ttl( )
  - ddns_use_option81( )
  - deny_bootp( )
  - email_list( )
  - enable_ddns( )
  - enable_dhcp_thresholds( )
  - enable_email_warnings( )
  - enable_snmp_warnings( )
  - extensible_attributes( )
  - fixed_address_templates( )
  - high_water_mark( )
  - ignore_dhcp_option_list_request( )
  - lease_scavenge_time( )
  - low_water_mark( )
  - members( )
  - name( )
  - netmask( )
  - nextserver( )
  - options( )
  - override_ddns_domainname( )
  - override_ddns_ttl( )
  - override_lease_scavenge_time( )
  - override_update_dns_on_lease_renewal( )
  - pxe_lease_time( )
  - range_templates( )
  - update_dns_on_lease_renewal( )
  - recycle_leases( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::NetworkTemplate - DHCP Network Template object

DESCRIPTION

You can use a template to create networks in a quick and consistent way. Networks created from a network template inherit all the properties defined in the network template, except for the comment and netmask if they are defined in the network.

CONSTRUCTOR

```
my $network_template = Infoblox::DHCP::NetworkTemplate->new(
        name                          => $string,                                    #Required
        allow_any_netmask              => "true" | "false",
#Optional / Default is "false"
```

```
        authority                       => "true" | "false",
#Optional / Default is "false"
        auto_create_reversezone         => "true" | "false"
#Optional / Default is "false"
        bootfile                        => $filename | undef,
#Optional / Default is undefined
        bootserver                      => $ipv4addr | $fqdn | undef,
#Optional / Default is undefined
        comment                         => $string,                                     #Optional
/ Default is empty
        ddns_domainname                 => $string | undef,
#Optional / Default is undefined
        ddns_generate_hostname          => "true" | "false" | undef,
#Optional / Default is undefined
        ddns_update_fixed_addresses     => "true" | "false" | undef,
#Optional / Default is undefined
        ddns_server_always_updates      => "true" | "false",
#Optional / Default is "true"
        ddns_ttl                        => $num,                                        #Optional
/ Default is 0
        ddns_use_option81               => "true" | "false" | undef,
#Optional / Default is undefined
        deny_bootp                      => "true" | "false" | undef,
#Optional / Default undefined
        email_list                      => [ "admin\@test.com",...],
#Optional / Default is undefined
        enable_ddns                     => "true" | "false" | undef,
#Optional / Default is undefined
        enable_dhcp_thresholds          => "true" | "false",
#Optional / Default is undefined
        enable_email_warnings           => "true" | "false",
#Optional / Default is undefined
        enable_snmp_warnings            => "true" | "false",
#Optional / Default is undefined
        extensible_attributes           => { $string => $string | $num, $string => [ $string | $num, ...
], ... }, #Optional / Default is undefined
        fixed_address_templates         => [$FixedAddrTemplate1,$FixedAddrTemplate2,...]
#Optional / Default is empty
        high_water_mark                 => $string,                                     #Optional
/ Default is empty
        ignore_dhcp_option_list_request => "true" | "false" | undef,
#Optional / Default is undefined
        lease_scavenge_time             => $number,                                     #Optional
/ Default is -1
        low_water_mark                  => $string,                                     #Optional
/ Default is empty
        members                         => [$Member1,$Member2,...],
#Optional / Default is empty
        netmask                         => $netmask,                                    #Optional
/ Default is /8
        nextserver                      => $ipv4addr | $fqdn | undef,
#Optional / Default is undefined
        options                         => [$Option1,$Option2,...]
#Optional / Default is empty
        override_ddns_ttl               => "true" | "false",
#Optional / Default value is "false"
        override_ddns_domainname        => "true" | "false",
#Optional / Default value is "false"
        override_lease_scavenge_time    => "true" | "false",
#Optional / Default value is "false"
        override_update_dns_on_lease_renewal  => "true" | "false",
#Optional / Default value is "false"
        pxe_lease_time                  => $num | undef,
#Optional / Default is undefined
        range_templates                 => [$RangeTemplate1,$RangeTemplate2,...]        ,
#Optional / Default is empty
        recycle_leases                  => "true" | "false" | undef,          #Optional / Default is
undefined
        update_dns_on_lease_renewal     => "true" | "false",
#Optional / Default value is "false"
          );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a DHCP Network Template object.

Infoblox::Session->add( )

Use this method to add a network template object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $network_template = Infoblox::DHCP::NetworkTemplate->new(
       name              => "custom_network_template",
       allow_any_netmask => "true",
);

# Submit for adding network template
my $response = $session->add( $network_template );
```

Infoblox::Session->get( )

Use this method to retrieve the matching network template objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP Network template object:

 name - Required. The name of network template.
```

Example

```
 my  @result_array = $session->get(
      object  => "Infoblox::DHCP::NetworkTemplate" ,
      name    => "custom_network_template"
 );
```

Infoblox::Session->search( )

Use this method to retrieve all the matching network template objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP Network Template object:

 name    - Optional. The name of the network template (regular
           expression).
 comment - Optional. The comment of the network template (regular
           expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
 my  @result_array = $session->search(
      object  => "Infoblox::DHCP::NetworkTemplate" ,
      name    => "custom_network_template"
      comment => ".*",
 );

 # search all Network Templates with a given extensible attribute 'Site'
 my @retrieved_objs = $session->search(
   object => "Infoblox::DHCP::NetworkTemplate",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a network template object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify enable_ddns method.
$object->enable_ddns("true");
# Submit modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
```

```
my  @result_array = $session->get(

      object => "Infoblox::DHCP::NetworkTemplate" ,
      name   => "custom_network_template"
 );

# find the desired object from retrieved list.
my $desired_network_template = $result_array[0];

# Submit for removal
my $response = $session->remove( $desired_network_template );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP Network template object

allow_any_netmask( )

Use this method to set or retrieve the allow_any_netmask parameter of a DHCP network template object. If you set this parameter to "true", then the template allows any netmask. A user must specify a netmask when creating a network using this template. If you set this parameter to false, you must specify the "netmask" property for the network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the allow_any_netmask parameter or specify "false" if you want to specify a particular netmask. The default value is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get allow_any_netmask
my $allow_any_netmask = $network_template->allow_any_netmask()
# Modify allow_any_netmask
$network_template->allow_any_netmask("true");
```

authority( )

Use this method to set or retrieve the authority flag of a DHCP network template object. This flag specifies if a DHCP server is authoritative for the network.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the authority flag or "false" to deactivate/unset it. The default value is undefined which indicates that this attribute inherits the upper-level setting.

```
=item B<Returns>
```

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get authority
my $authority = $network_template->authority()
# Modify authority
$network_template->authority("true");
```

auto_create_reversezone( )

Use this method to set or retrieve the auto_create_reversezone flag of a DHCP network template object. This flag indicates whether the appliance must automatically create the corresponding reverse zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to create a reverse zone automatically or "false" to disable the automatic creation of a reverse zone.The default value for this field is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get auto_create_reversezone
my $auto_create_reversezone = $network_template->auto_create_reversezone();
# Modify auto_create_reversezone
$network_template->auto_create_reversezone("true");
```

bootfile( )

Use this method to set or retrieve the bootfile attribute of a DHCP network template object. You can configure the DHCP server to support clients that use the boot file name option in their DHCPREQUEST messages. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::NetworkTemplate->bootserver() and Infoblox::DHCP::NetworkTemplate->nextserver().

Parameter

The name, in string format, of the file that the client must download. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get bootfile
my $bootfile = $network_template->bootfile();
# Modify bootfile
$network_template->bootfile("bootfile1");
#Un-override bootfile
$network_template->bootfile(undef);
```

bootserver( )

Use this method to set or retrieve the boot server attribute of a DHCP network template object. You can specify the name of the boot server on which the boot file is stored. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::NetworkTemplate->bootfile() and Infoblox::DHCP::NetworkTemplate->nextserver().

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get bootserver
my $bootserver = $network_template->bootserver();
# Modify bootserver
$network_template->bootserver("abc.domain.com");
#Un-override bootserver
$network_template->bootserver(undef);
```

comment( )

Use this method to set or retrieve a descriptive comment about a DHCP network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network template in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $network_template->comment();
# Modify comment
$network_template->comment("add a network");
```

ddns_domainname( )

Use this method to set or retrieve the ddns_domainname value of a DHCP Network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_dddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

The dynamic DNS domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_domainname
my $ddns_domainname = $network_template->ddns_domainname();
#Modify ddns_domainname
$network_template->ddns_domainname("test_domain.com");
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname flag of a DHCP network template object. Set this flag to allow only the DHCP server to generate a host name. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to generate hostname or "false" to deactivate/unset it. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_generate_hostname
my $ddns_generate_hostname = $network_template->ddns_generate_hostname()
# Modify ddns_generate_hostname
$network_template->ddns_generate_hostname("true");
#Un-override
$network_template->ddns_generate_hostname(undef);
```

ddns_update_fixed_addresses( )

Use this method to set or retrieve the ddns_update_fixed_addresses flag of a DHCP network template object. By default, the DHCP server does not update DNS when it allocates a fixed address to a client. You can configure the DHCP server to update the A and PTR records of a client with a fixed address. When this feature is enabled and the DHCP server adds A and PTR records for a fixed address, the DHCP server never discards the records. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to update DNS when it allocates a fixed address to a client or "false" to deactivate/unset it. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_update_fixed_addresses
my $ddns_update_fixed_addresses= $network_template->ddns_update_fixed_addresses()
# Modify ddns_update_fixed_addresses
$network_template->ddns_update_fixed_addresses("true");
#Un-override
$network_template->ddns_update_fixed_addresses(undef);
```

ddns_server_always_updates( )

Use this method to set or retrieve the ddns_server_always_updates flag of a DHCP network template object. Set this flag to allow the DHCP server only to update DNS, regardless of the requests from DHCP clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set this flag and allow the DHCP server only to update DNS or "false" to deactivate/unset the flag. The default value is true.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_server_always_updates
my $ddns_server_always_updates = $network_template->ddns_server_always_updates()
# Modify ddns_server_always_updates
$network_template->ddns_server_always_updates("true");
```

ddns_ttl( )

Use this method to set or retrieve the DNS update Time to Live (TTL) value of a DHCP network template object. The TTL value indicates the number of seconds the Infoblox appliance caches a DDNS update.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_ttl method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_ttl attribute to "false".

Note that when ddns_ttl is set to a defined value and override_dddns_ttl is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_ttl(12); $object->override_ddns_ttl("false"); will set override_ddns_ttl to "false", and the sequence $object->override_ddns_ttl("false"); $object->ddns_ttl(12); will result in override_ddns_ttl="true".

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the update is cached. Zero indicates that the update should not be cached. The default value is 0.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get DNS Update ttl
my $dns_ttl = $network_template->ddns_ttl();
#Modify DNS Update ttl
$network_template->ddns_ttl(1200);
```

ddns_use_option81( )

Use this method to set or retrieve the ddns_use_option81 flag of a DHCP network template object. Enable this method to define the settings for option 81 at the network template level. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable support for option 81 or "false" to disable support. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_use_option81
my $ddns_use_option81 = $network_template->ddns_use_option81()
# Modify ddns_use_option81
$network_template->ddns_use_option81("true");
# un-override ddns_use_option81
$network_template->ddns_use_option81(undef);
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp flag of a DHCP Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_bootp flag or "false" to deactivate/unset it. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $network->deny_bootp();
#Modify deny_bootp
$network->deny_bootp("true");
```

email_list( )

Use this method to set or retrieve the e-mail lists to which the appliance sends threshold e-mail messages.

If you enter e-mail addresses in the email_list, then the appliance sends DHCP threshold alarms to the e-mail addresses.

Parameter

The valid value is an array reference that contains string objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get email_list
my @email_list = $network_template->email_list();
#Modify email_list
```

```
$network_template->email_list( [ "admin\@test.com" ] );
```

enable_ddns( )

Use this method to set or retrieve the dynamic DNS updates flag of a DHCP network template object. The DHCP server can send DDNS updates to DNS servers in the same grid and to external DNS servers. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the dynamic DNS updates flag or "false" to deactivate/unset it. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_ddns
my $enable_ddns = $network_template->enable_ddns()
# Modify enable_ddns
$network_template->enable_ddns("true");
# Un-override enable_ddns
$network_template->enable_ddns(undef);
```

enable_dhcp_thresholds( )

Use this method to set or retrieve the DHCP thresholds of a network template object. These thresholds are the high and low watermarks. If DHCP range usage goes above or below these percentage values, the Infoblox appliance generates a syslog message and sends an SNMP notification.

Enable this flag to set DHCP threshold values.

Parameter

Specify "true" to set the enable_dhcp_thresholds flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcp_thresholds
my $enable_dhcp_thresholds = $network_template->enable_dhcp_thresholds();
#Modify enable_dhcp_thresholds
$network_template->enable_dhcp_thresholds("false");
```

enable_email_warnings( )

Use this method to send DHCP thresholds warnings through e-mail.

Enable this flag if you want the appliance to send DHCP threshold warnings to the "email_list" provided.

Parameter

+Specify "true" to set the enable_email_warnings flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_email_warnings
my $enable_email_warnings = $network_template->enable_email_warnings();
#Modify enable_email_warnings
$network_template->enable_email_warnings("false");
```

enable_snmp_warnings( )

Use this method to send DHCP threshold warnings through SNMP.

If the flag is enabled, then the appliance sends DHCP threshold warnings through SNMP messages.

Parameter

Specify "true" to set the enable_snmp_warnings flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_snmp_warnings
my $enable_snmp_warnings = $network_template->enable_snmp_warnings();
#Modify enable_snmp_warnings
$network_template->enable_snmp_warnings("false");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a Network Template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $network_template->extensible_attributes();
#Modify extensible attributes
$network_template->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter',
'Tom' ] });
```

fixed_address_templates( )

Use this method to set or retrieve the fixed address template assigned to this network template object. When you create a network based on a network template object with a fixed address template, the fixed addresses are created based on the associated fixed address template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference of Infoblox::DHCP::Template objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get fixed_address_templates
my $fixed_address_templates = $network_template->fixed_address_templates();
# Modify fixed_address_templates
my $tmpl = Infoblox::DHCP::Template->new(
   name => "custom_fixed_address_template"
);
$network_template->fixed_address_templates([$tmpl]);
```

high_water_mark( )

Use this method to set the percentage value for DHCP range usage above which the Infoblox appliance generates a syslog message and sends a warning.

When the percent of allocated addresses in a DHCP range exceeds the high_water_mark, the appliance generates a syslog message and sends a warning.

Parameter

The valid value is in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get high_water_mark
my $high_water_mark = $network_template->high_water_mark();
#Modify high_water_mark
$network_template->high_water_mark("80");
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the ignore_dhcp_option_list_request flag of a DHCP network template object. Set this flag to ignore option lists requested by clients and return all defined options at the network template level.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate/unset it. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $network_template->ignore_dhcp_option_list_request();
# Modify ignore_dhcp_option_list_request
$network_template->ignore_dhcp_option_list_request("false");
# Un-override ignore_dhcp_option_list_request
$network_template->ignore_dhcp_option_list_request(undef);
```

lease_scavenge_time( )

Use this method to set or retrieve the lease_scavenge_time value of a Network Template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_lease_scavenge_time method to "true". Setting the parameter to undefined causes the appliance to use the default from the upper level and automatically resets the override_lease_scavenge_time attribute to "false".

Note that when lease_scavenge_time is set to a defined value and override_lease_scavenge_time is set to "false", the last operation takes precedence. Thus the sequence $object->lease_scavenge_time(604800); $object->override_lease_scavenge_time("false"); will set override_lease_scavenge_time to "false", and the sequence $object->override_lease_scavenge_time("false"); $object->lease_scavenge_time(604800); will result in override_lease_scavenge_time="true".

Parameter

An integer that specifies the period of time, in seconds, that free and backup leases remain in the database before they are automatically deleted. To disable lease scavenging, set the parameter to -1. The minimum positive value must be greater than 86400 seconds (1 day). The default value of this parameter is -1, which means that lease scavenging is disabled.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lease_scavenge_time
my $lease_scavenge = $network_template->lease_scavenge_time();
#Modify lease_scavenge_time
$network_template->lease_scavenge_time(30 * 24 * 60 * 60);
```

low_water_mark( )

Use this method to set the percentage value for DHCP range usage below which the Infoblox appliance generates a syslog message and sends a warning.

When the percent of allocated addresses in a DHCP range goes below the low_water_mark, the appliance generates a syslog message and sends a warning.

Parameter

The valid value is in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get low_water_mark
my $low_water_mark = $network_template->low_water_mark();
#Modify low_water_mark
$network_template->low_water_mark("10");
```

members( )

Use this method to set or retrieve the member or Microsoft (r) server that serves DHCP for the network created from the network template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Member or Infoblox::DHCP::MSServer objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $member1 = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
);

# Get members
my $members = $network_template->members();
# Modify members
$network_template->members([$member1]);
```

name( )

Use this method to set or retrieve the description of the name of the DHCP network template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the network template in string format with a maximum of 64 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $network_template->name();
```

```
#Modify name
$network_template->name("custom_network_template");
```

netmask( )

Use this method to set or retrieve the netmask attribute of a DHCP network template object. If "allow_any_netmask" is set to "false", then this attribute must be specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A valid subnet mask. The default value is "255.0.0.0".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get netmask
my $netmask = $network_template->netmask();
#Set netmask
$nework_template->allow_any_netmask( "false" );
my $netmask = $network_template->netmask("255.0.0.0");
```

nextserver( )

Use this method to set or retrieve the nextserver attribute of a DHCP network template object. You can specify the name and/or IP address of the next server to which the host needs to connect in order to boot. This setting overrides the upper-level settings. The default value is undefined which indicates that this attribute inherits the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::NetworkTemplate->bootfile() and Infoblox::DHCP::NetworkTemplate->bootserver().

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get nextserver
my $nextserver = $network_template->nextserver();
# Modify nextserver
$network_template->nextserver("blue.domain.com");
#Un-override nextserver
$network_template->nextserver(undef);
```

options( )

Use this method to set or retrieve the options value of a network template object. DHCP options provide network configuration settings and various services available on the network for DHCP. This setting overrides the upper-level setting. The default value is undefined which indicates that this attribute inherit the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $network_template->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name  => "lease-time",
    value => 50
);

$network_template->options([$option1]);
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP network template object is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$network_template->override_ddns_domainname( );
#Modifying override_ddns_domainname
$network_template->override_ddns_domainname("true");
```

override_ddns_ttl( )

The override_ddns_ttl attribute controls whether the ddns_ttl value in the DHCP network template object is used, instead of the grid default.

The override_ddns_ttl attribute can be specified explicitly. It is also set implicitly when ddns_ttl is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_ttl. Set the parameter to "false" to inherit the grid-level setting for ddns_ttl.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_ttl
my $override_ddns_ttl=$network_template->override_ddns_ttl( );
#Modifying override_ddns_ttl
$network_template->override_ddns_ttl("true");
```

override_lease_scavenge_time( )

The override_lease_scavenge_time attribute controls whether the lease_scavenge_time value in the network template is used, instead of the upper-level default.

The override_lease_scavenge_time attribute can be specified explicitly. It is also set implicitly when lease_scavenge_time is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for lease_scavenge_time. Set the parameter to "false" to inherit the setting for lease_scavenge_time from the upper level.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_lease_scavenge_time
my $override_lease_scavenge_time=$network_template->override_lease_scavenge_time( );
#Modifying override_lease_scavenge_time
$network_template->override_lease_scavenge_time("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the DHCP network template object is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$network_template->override_update_dns_on_lease_renewal(
);
#Modifying override_update_dns_on_lease_renewal
$network_template->override_update_dns_on_lease_renewal("true");
```

pxe_lease_time( )

Use this method to set or retrieve the PXE lease time value of a DHCP network template object. Some hosts use PXE (Preboot Execution Environment) to boot remotely from a server. To better manage your IP resources, set a different lease time for PXE boot requests. You can configure the DHCP server to allocate an IP address with a shorter lease time to hosts that send PXE boot requests, so IP addresses are not leased longer than necessary. This setting overrides the upper-level settings. The default value is undefined, which indicates that this attribute inherits the upper-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the update is cached. Zero indicates that the update should not be cached. The default value is undefined

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get PXE Lease Time
my $pxe_lease_time = $network_template->pxe_lease_time();
#Modify PXE Lease Time
```

```
$network_template->pxe_lease_time(1100);
#Un-override PXE Lease Time
$network_template->pxe_lease_time(undef);
```

range_templates( )

Use this method to set or retrieve the IP address range template assigned to this network template object. When you create a network based on a network template object with a range template, the IP address ranges are created based on the associated IP address range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference of Infoblox::DHCP::Template objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get range_templates
my $range_templates = $network_template->range_templates();
#Modify range_templates
my $tmpl = Infoblox::DHCP::Template->new(
   name => "custom_range_template"
);
$network_template->range_templates([ $tmpl ]);
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$network_template->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$network_template->update_dns_on_lease_renewal("true");
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag at the network template level.

If the flag is enabled, the leases are kept in the recycle bin until one week after expiration. When the flag is disabled, the leases are irrecoverably deleted.

Parameter

Specify "true" to enable the recycle_leases flag or "false" to disable it. The default value is undefined, which indicates that this attribute inherit the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $network_template->recycle_leases();
#Modify recycle_leases
$network_template->recycle_leases("false");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. Also, this sample code includes error handling for the operations.

#Preparation prior to a DHCP network template insertion

```
#PROGRAM STARTS: Include all the modules that will be used

use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $host_ip = "192.168.1.2";
my $host_name = "infoblox.localdomain";

my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $memberdhcp = Infoblox::DHCP::Member->new(
            name     => $host_name,
            ipv4addr => $host_ip
           );
```

#Add network template object

```
#Create a DHCP Network template object

my $network_template = Infoblox::DHCP::NetworkTemplate->new(
                                name                   => "custom_network_template",
                                netmask                => "255.0.0.0",
                                comment                => "add network template",
                                members                => [ $memberdhcp ],
                                authority              => "true",
                                enable_ddns            => "true",
                                pxe_lease_time         => "3600",
                                bootserver             => "someserver.com",
                                nextserver             => "3.3.3.3",
                                bootfile               => "bootfile.com",
                                auto_create_reversezone => "true",
                                );

my $response = $session->add($network_template)
       or die("Add Network template failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network template added successfully\n";
```

#Get and modify a network template attribute

```
#Get Network template through the session
my  @retrieved_objs = $session->get(
           object => "Infoblox::DHCP::NetworkTemplate",
           name   => "custom_network_template"
        );
```

```
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get Network template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network template found at least 1 matching entry\n";
```

#Search a network template attribute

```
#Search Network template through the session
my  @retrieved_objs = $session->search(
           object => "Infoblox::DHCP::NetworkTemplate",
           name   => "custom_network_template"
        );

my $object = $retrieved_objs[0];

unless ($object) {
    die("Search Network template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search Network template found at least 1 matching entry\n";

#Modify attributes of the specified Network Template

$object->enable_ddns("true"); #Apply the changes

#Un-override bootfile
$object->bootfile(undef);

#Un-override bootserver
$object->bootserver(undef);

#Un-override nextserver
$object->nextserver(undef);

$session->modify($object)
    or die("Modify Network template attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "Network template object modified successfully \n";
```

#Remove a network template object

```
#Get Network Template object through the session

 my @retrieved_objs = $session->get(
           object  => "Infoblox::DHCP::NetworkTemplate",
           name    => "custom_network_template"
       );

   my $object = $retrieved_objs[0];

   unless ($object) {
    die("Get Network Template object failed: ",
        $session->status_code() . ":" . $session->status_detail());
    }

print "Get Network Template object found at least 1 matching entry\n";

#Submit the object for removal

$session->remove($object)
    or die("Remove Network Template failed: ",
        $session->status_code() . ":" . $session->status_detail());

print "Network Template removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::Member, Infoblox::DHCP::Option, Infoblox::DHCP::Network, Infoblox::DHCP::RangeTemplate, Infoblox::DHCP::FixedAddrTemplate, Infoblox::DHCP::Template, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Option - DHCP Option object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DHCP->options( )
  - Infoblox::Grid::Member::DHCP->options( )
  - Infoblox::DHCP::Network->options( )
  - Infoblox::DHCP::NetworkTemplate->options( )
  - Infoblox::DHCP::SharedNetwork->options( )
  - Infoblox::DHCP::FixedAddr->options( )
  - Infoblox::DHCP::FixedAddrTemplate->options( )
  - Infoblox::DHCP::Range->options( )
  - Infoblox::DHCP::RangeTemplate->options( )
  - Infoblox::DHCP::Filter::Option->option_list( )
- METHODS
  - name( )
  - num( )
  - value( )
  - vendor_class( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Option - DHCP Option object.

DESCRIPTION

An Option sets the value of a DHCP option that has been defined in an option space. DHCP options describe network configuration settings and various services available on the network. These options occur as variable-length fields at the end of DHCP messages.

CONSTRUCTOR

```
 my $option = Infoblox::DHCP::Option->new(
     value        => $num | $string |"true"|"false", #Required
     name         => $string,                        #Optional - One of name or num must be specified /
Default is undefined
     num          => $num,                           #Optional - One of name or num must be specified /
Default is undefined
     vendor_class => $string,                        #Optional / Default Option Space name is "DHCP"
 );
```

MODULE METHODS

The following functions can be applied to a DHCP Option object.

Infoblox::Grid::DHCP->options( )

Use this function to specify DHCP options at the grid level on the Infoblox appliance. See Infoblox::Grid::DHCP->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option1 = Infoblox::DHCP::Option->new(
              name  => "lease-time",
              value => "50"
);
# Configure a DHCP option for a grid
my $response = $Grid_DHCP->options([$option1]);
```

Infoblox::Grid::Member::DHCP->options( )

Use this function to specify DHCP options at the grid member level on the Infoblox appliance. See Infoblox::Grid::Member::DHCP->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option2 = Infoblox::DHCP::Option->new(
              name  => "domain-name",
              value => "domainname.com"
);
# Configure a DHCP option for a grid member
my $response = $Grid_Member_DHCP->options([$option2]);
```

Infoblox::DHCP::Network->options( )

Use this function to specify DHCP options at the network level on the Infoblox appliance. See Infoblox::DHCP::Network->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option3 = Infoblox::DHCP::Option->new(
              name  => "broadcast",
              value => "255.0.0.0"
);
# Configure a DHCP option for a network
my $response = $Network_DHCP->options([$option3]);
```

Infoblox::DHCP::NetworkTemplate->options( )

Use this function to specify DHCP options at the network template level on the Infoblox appliance. See Infoblox::DHCP::NetworkTemplate->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option4 = Infoblox::DHCP::Option->new(
              name  => "broadcast",
              value => "2"
);
#Construct another DHCP Option object
my $option5 = Infoblox::DHCP::Option->new(
              name  => "routers",
              value => "1,2"
);
# Configure a DHCP options for a network template
my $response = $Network_Template_DHCP->options([$option4,$option5]);
```

Infoblox::DHCP::SharedNetwork->options( )

Use this function to specify DHCP options at the shared network level on the Infoblox appliance. See Infoblox::DHCP::SharedNetwork->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option6 = Infoblox::DHCP::Option->new(
              name  => "subnet-mask",
              value => "255.255.0.0"
);
# Configure a DHCP option for a shared network
my $response = $Shared_Network_DHCP->options([$option6]);
```

Infoblox::DHCP::FixedAddr->options( )

Use this function to specify DHCP options at the fixed address level on the Infoblox appliance. See Infoblox::DHCP::FixedAddr->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option7 = Infoblox::DHCP::Option->new(
              name  => "host-name",
              value => "host1.com"
);
# Configure a DHCP option for a fixed address
my $response = $Fixed_Address_DHCP->options([$option7]);
```

Infoblox::DHCP::FixedAddrTemplate->options( )

Use this function to specify DHCP options at the fixed address template level on the Infoblox appliance. See Infoblox::DHCP::FixedAddrTemplate->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option8 = Infoblox::DHCP::Option->new(
              name  => "broadcast",
              value => "2"
);
#Construct another DHCP Option object
my $option9 = Infoblox::DHCP::Option->new(
              name  => "routers",
              value => "1,2"
);
# Configure a DHCP options for a fixed address template
my $response = $Fixed_Address_Template_DHCP->options([$option8,$option9]);
```

Infoblox::DHCP::Range->options( )

Use this function to specify DHCP options at the DHCP range level on the Infoblox appliance. See Infoblox::DHCP::Range->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option10 = Infoblox::DHCP::Option->new(
              name  => "swap-server",
              value => "1.1.1.1"
);
# Configure a DHCP option for a DHCP range
my $response = $DHCP_Range->options([$option10]);
```

Infoblox::DHCP::RangeTemplate->options( )

Use this function to specify DHCP options at the range template level on the Infoblox appliance. See Infoblox::DHCP::RangeTemplate->options() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option11 = Infoblox::DHCP::Option->new(
              name  => "broadcast",
              value => "2"
);
#Construct another DHCP Option object
my $option12 = Infoblox::DHCP::Option->new(
              name  => "routers",
              value => "1,2"
);
# Configure a DHCP options for a range template
my $response = $Range_Template_DHCP->options([$option11,$option12]);
```

Infoblox::DHCP::Filter::Option->option_list( )

Use this function to specify DHCP options at the filter level on the Infoblox appliance. See Infoblox::DHCP::Filter::Option->option_list() for parameters and return values.

Example

```
#Construct a DHCP Option object
my $option13 = Infoblox::DHCP::Option->new(
              name  => "pop-server",
              value => "10.10.10.10"
);
# Configure a DHCP Option for a filter
my $response = $Filter_Option->option_list([$option13]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP option.

name( )

Use this method to set or retrieve the name of the DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the name of the DHCP option in string format. Valid values are

```
  "subnet-mask",                              - DHCP Option   1                - ip-address
  "time-offset",                              - DHCP Option   2                - 32-bit signed integer
  "routers",                                  - DHCP Option   3                - array of ip-address |
array of offsets (for templates)
  "time-servers",                             - DHCP Option   4                - array of ip-address
  "ien116-name-servers",                      - DHCP Option   5                - array of ip-address
  "nameservers",                              - DHCP Option   6                - array of ip-address
  "log-servers",                              - DHCP Option   7                - array of ip-address
  "cookie-servers",                           - DHCP Option   8                - array of ip-address
  "lpr-servers",                              - DHCP Option   9                - array of ip-address
  "impress-servers",                          - DHCP Option  10                - array of ip-address
  "resource-location-servers",                - DHCP Option  11                - array of ip-address
  "host-name",                                - DHCP Option  12                - string
  "boot-size",                                - DHCP Option  13                - 16-bit unsigned integer
  "merit-dump",                               - DHCP Option  14                - text
  "domain-name",                              - DHCP Option  15                - text
  "swap-server",                              - DHCP Option  16                - ip-address
  "root-path",                                - DHCP Option  17                - text
  "extensions-path",                          - DHCP Option  18                - text
  "ip-forwarding",                            - DHCP Option  19                - boolean
  "non-local-source-routing",                 - DHCP Option  20                - boolean
  "policy-filter",                            - DHCP Option  21                - array of ip-address |
ip-address
  "max-dgram-reassembly",                     - DHCP Option  22                - 16-bit unsigned integer
  "default-ip-ttl",                           - DHCP Option  23                - 8-bit unsigned integer
  "path-mtu-aging-timeout",                   - DHCP Option  24                - 32-bit unsigned integer
  "path-mtu-plateau-table",                   - DHCP Option  25                - array of 16-bit
unsigned integer
  "interface-mtu",                            - DHCP Option  26                - 16-bit unsigned integer
  "all-subnets-local",                        - DHCP Option  27                - boolean
  "broadcast",                                - DHCP Option  28                - ip-address | offset (for
templates)
  "perform-mask-discovery",                   - DHCP Option  29                - boolean
  "mask-supplier",                            - DHCP Option  30                - boolean
  "router-discovery",                         - DHCP Option  31                - boolean
  "router-solicitation-address",              - DHCP Option  32                - ip-address
  "static-routes",                            - DHCP Option  33                - array of ip-address |
ip-address
  "trailer-encapsulation",                    - DHCP Option  34                - boolean
  "arp-cache-timeout",                        - DHCP Option  35                - 32-bit unsigned integer
  "ieee802-3-encapsulation",                  - DHCP Option  36                - boolean
  "default-tcp-ttl",                          - DHCP Option  37                - 8-bit unsigned integer
  "tcp-keepalive-interval",                   - DHCP Option  38                - 32-bit unsigned integer
  "tcp-keepalive-garbage",                    - DHCP Option  39                - boolean
  "nis-domain",                               - DHCP Option  40                - text
  "nis-servers",                              - DHCP Option  41                - array of ip-address
  "ntp-servers",                              - DHCP Option  42                - array of ip-address
  "vendor-encapsulated-options",              - DHCP Option  43                - string
  "netbios-name-servers",                     - DHCP Option  44                - array of ip-address
  "netbios-dd-server",                        - DHCP Option  45                - array of ip-address
  "netbios-node-type",                        - DHCP Option  46                - 8-bit unsigned
integer(1,2,4,8)
  "netbios-scope",                            - DHCP Option  47                - text
  "font-servers",                             - DHCP Option  48                - array of ip-address
  "x-display-manager",                        - DHCP Option  49                - array of ip-address
  "dhcp-requested-address",                   - DHCP Option  50                - ip-address
  "lease-time",                               - DHCP Option  51                - 32-bit unsigned integer
  "dhcp-option-overload",                     - DHCP Option  52                - 8-bit unsigned integer
  "dhcp-message-type",                        - DHCP Option  53                - 8-bit unsigned integer
  "dhcp-server-identifier",                   - DHCP Option  54                - ip-address
  "dhcp-parameter-request-list",              - DHCP Option  55                - array of 8-bit unsigned
integer
  "dhcp-message",                             - DHCP Option  56                - text
  "dhcp-max-message-size",                    - DHCP Option  57                - 16-bit unsigned integer
  "dhcp-renewal-time",                        - DHCP Option  58                - 32-bit unsigned integer
  "dhcp-rebinding-time",                      - DHCP Option  59                - 32-bit unsigned integer
  "vendor-class-identifier",                  - DHCP Option  60                - string
  "dhcp-client-identifier",                   - DHCP Option  61                - string
  "nwip-domain",                              - DHCP Option  62                - string
  "nwip-suboptions",                          - DHCP Option  63                - string
  "nisplus-domain",                           - DHCP Option  64                - text
  "nisplus-servers",                          - DHCP Option  65                - array of ip-address
  "tftp-server-name",                         - DHCP Option  66                - text
  "bootfile-name",                            - DHCP Option  67                - text
  "mobile-ip-home-agent",                     - DHCP Option  68                - array of ip-address
  "smtp-server",                              - DHCP Option  69                - array of ip-address
  "pop-server",                               - DHCP Option  70                - array of ip-address
  "nntp-server",                              - DHCP Option  71                - array of ip-address
  "www-server",                               - DHCP Option  72                - array of ip-address
  "finger-server",                            - DHCP Option  73                - array of ip-address
  "irc-server",                               - DHCP Option  74                - array of ip-address
  "streettalk-server",                        - DHCP Option  75                - array of ip-address
  "streettalk-directory-assistance-server", - DHCP Option  76                - array of ip-address
  "user-class",                               - DHCP Option  77                - text
  "slp-directory-agent",                      - DHCP Option  78                - boolean | array of ip-
address
  "slp-service-scope",                        - DHCP Option  79                - boolean | text
  "option-80",                                - DHCP Option  80                - custom defined
  "fqdn",                                     - DHCP Option  81                - string
```

"relay-agent-information",	- DHCP Option 82	- string
"option-83",	- DHCP Option 83	- custom defined
"option-84",	- DHCP Option 84	- custom defined
"nds-servers",	- DHCP Option 85	- array of ip-address
"nds-tree-name",	- DHCP Option 86	- string
"nds-context",	- DHCP Option 87	- string
"option-88",	- DHCP Option 88	- custom defined
"bcms-controller-address",	- DHCP Option 89	- array of ip-address
"option-90",	- DHCP Option 90	- custom defined
"client-last-transaction-time",	- DHCP Option 91	- 32-bit unsigned integer
"associated-ip",	- DHCP Option 92	- array of ip-address
"option-93",	- DHCP Option 93	- custom defined
"option-94",	- DHCP Option 94	- custom defined
"option-95",	- DHCP Option 95	- custom defined
"option-96",	- DHCP Option 96	- custom defined
"option-97",	- DHCP Option 97	- custom defined
"uap-servers",	- DHCP Option 98	- text
"option-99",	- DHCP Option 99	- custom defined
"option-100",	- DHCP Option 100	- custom defined
"option-101",	- DHCP Option 101	- custom defined
"option-102",	- DHCP Option 102	- custom defined
"option-103",	- DHCP Option 103	- custom defined
"option-104",	- DHCP Option 104	- custom defined
"option-105",	- DHCP Option 105	- custom defined
"option-106",	- DHCP Option 106	- custom defined
"option-107",	- DHCP Option 107	- custom defined
"option-108",	- DHCP Option 108	- custom defined
"option-109",	- DHCP Option 109	- custom defined
"option-110",	- DHCP Option 110	- custom defined
"option-111",	- DHCP Option 111	- custom defined
"netinfo-server-address",	- DHCP Option 112	- array of ip-address
"netinfo-server-tag",	- DHCP Option 113	- text
"default-url",	- DHCP Option 114	- text
"option-115",	- DHCP Option 115	- custom defined
"option-116",	- DHCP Option 116	- custom defined
"option-117",	- DHCP Option 117	- custom defined
"subnet-selection",	- DHCP Option 118	- string
"domain-search",	- DHCP Option 119	- domain-list
"option-120",	- DHCP Option 120	- custom defined
"option-121",	- DHCP Option 121	- custom defined
"option-122",	- DHCP Option 122	- custom defined
"option-123",	- DHCP Option 123	- custom defined
"vivco",	- DHCP Option 124	- string
"vivso",	- DHCP Option 125	- string
"option-126",	- DHCP Option 126	- custom defined
"option-127",	- DHCP Option 127	- custom defined
"option-128",	- DHCP Option 128	- custom defined
"option-129",	- DHCP Option 129	- custom defined
"option-130",	- DHCP Option 130	- custom defined
"option-131",	- DHCP Option 131	- custom defined
"option-132",	- DHCP Option 132	- custom defined
"option-133",	- DHCP Option 133	- custom defined
"option-134",	- DHCP Option 134	- custom defined
"option-135",	- DHCP Option 135	- custom defined
"option-136",	- DHCP Option 136	- custom defined
"option-137",	- DHCP Option 137	- custom defined
"option-138",	- DHCP Option 138	- custom defined
"option-139",	- DHCP Option 139	- custom defined
"option-140",	- DHCP Option 140	- custom defined
"option-141",	- DHCP Option 141	- custom defined
"option-142",	- DHCP Option 142	- custom defined
"option-143",	- DHCP Option 143	- custom defined
"option-144",	- DHCP Option 144	- custom defined
"option-145",	- DHCP Option 145	- custom defined
"option-146",	- DHCP Option 146	- custom defined
"option-147",	- DHCP Option 147	- custom defined
"option-148",	- DHCP Option 148	- custom defined
"option-149",	- DHCP Option 149	- custom defined
"option-150",	- DHCP Option 150	- custom defined
"option-151",	- DHCP Option 151	- custom defined
"option-152",	- DHCP Option 152	- custom defined
"option-153",	- DHCP Option 153	- custom defined
"option-154",	- DHCP Option 154	- custom defined
"option-155",	- DHCP Option 155	- custom defined
"option-156",	- DHCP Option 156	- custom defined
"option-157",	- DHCP Option 157	- custom defined
"option-158",	- DHCP Option 158	- custom defined
"option-159",	- DHCP Option 159	- custom defined
"option-160",	- DHCP Option 160	- custom defined
"option-161",	- DHCP Option 161	- custom defined
"option-162",	- DHCP Option 162	- custom defined
"option-163",	- DHCP Option 163	- custom defined
"option-164",	- DHCP Option 164	- custom defined
"option-165",	- DHCP Option 165	- custom defined
"option-166",	- DHCP Option 166	- custom defined
"option-167",	- DHCP Option 167	- custom defined
"option-168",	- DHCP Option 168	- custom defined
"option-169",	- DHCP Option 169	- custom defined
"option-170",	- DHCP Option 170	- custom defined
"option-171",	- DHCP Option 171	- custom defined
"option-172",	- DHCP Option 172	- custom defined
"option-173",	- DHCP Option 173	- custom defined
"option-174",	- DHCP Option 174	- custom defined
"option-175",	- DHCP Option 175	- custom defined
"option-176",	- DHCP Option 176	- custom defined
"option-177",	- DHCP Option 177	- custom defined
"option-178",	- DHCP Option 178	- custom defined

```
"option-179",                    - DHCP Option 179          - custom defined
"option-180",                    - DHCP Option 180          - custom defined
"option-181",                    - DHCP Option 181          - custom defined
"option-182",                    - DHCP Option 182          - custom defined
"option-183",                    - DHCP Option 183          - custom defined
"option-184",                    - DHCP Option 184          - custom defined
"option-185",                    - DHCP Option 185          - custom defined
"option-186",                    - DHCP Option 186          - custom defined
"option-187",                    - DHCP Option 187          - custom defined
"option-188",                    - DHCP Option 188          - custom defined
"option-189",                    - DHCP Option 189          - custom defined
"option-190",                    - DHCP Option 190          - custom defined
"option-191",                    - DHCP Option 191          - custom defined
"option-192",                    - DHCP Option 192          - custom defined
"option-193",                    - DHCP Option 193          - custom defined
"option-194",                    - DHCP Option 194          - custom defined
"option-195",                    - DHCP Option 195          - custom defined
"option-196",                    - DHCP Option 196          - custom defined
"option-197",                    - DHCP Option 197          - custom defined
"option-198",                    - DHCP Option 198          - custom defined
"option-199",                    - DHCP Option 199          - custom defined
"option-200",                    - DHCP Option 200          - custom defined
"option-201",                    - DHCP Option 201          - custom defined
"option-202",                    - DHCP Option 202          - custom defined
"option-203",                    - DHCP Option 203          - custom defined
"option-204",                    - DHCP Option 204          - custom defined
"option-205",                    - DHCP Option 205          - custom defined
"option-206",                    - DHCP Option 206          - custom defined
"option-207",                    - DHCP Option 207          - custom defined
"option-208",                    - DHCP Option 208          - custom defined
"option-209",                    - DHCP Option 209          - custom defined
"option-210",                    - DHCP Option 210          - custom defined
"option-211",                    - DHCP Option 211          - custom defined
"option-212",                    - DHCP Option 212          - custom defined
"option-213",                    - DHCP Option 213          - custom defined
"option-214",                    - DHCP Option 214          - custom defined
"option-215",                    - DHCP Option 215          - custom defined
"option-216",                    - DHCP Option 216          - custom defined
"option-217",                    - DHCP Option 217          - custom defined
"option-218",                    - DHCP Option 218          - custom defined
"option-219",                    - DHCP Option 219          - custom defined
"option-220",                    - DHCP Option 220          - custom defined
"option-221",                    - DHCP Option 221          - custom defined
"option-222",                    - DHCP Option 222          - custom defined
"option-223",                    - DHCP Option 223          - custom defined
"option-224",                    - DHCP Option 224          - custom defined
"option-225",                    - DHCP Option 225          - custom defined
"option-226",                    - DHCP Option 226          - custom defined
"option-227",                    - DHCP Option 227          - custom defined
"option-228",                    - DHCP Option 228          - custom defined
"option-229",                    - DHCP Option 229          - custom defined
"option-230",                    - DHCP Option 230          - custom defined
"option-231",                    - DHCP Option 231          - custom defined
"option-232",                    - DHCP Option 232          - custom defined
"option-233",                    - DHCP Option 233          - custom defined
"option-234",                    - DHCP Option 234          - custom defined
"option-235",                    - DHCP Option 235          - custom defined
"option-236",                    - DHCP Option 236          - custom defined
"option-237",                    - DHCP Option 237          - custom defined
"option-238",                    - DHCP Option 238          - custom defined
"option-239",                    - DHCP Option 239          - custom defined
"option-240",                    - DHCP Option 240          - custom defined
"option-241",                    - DHCP Option 241          - custom defined
"option-242",                    - DHCP Option 242          - custom defined
"option-243",                    - DHCP Option 243          - custom defined
"option-244",                    - DHCP Option 244          - custom defined
"option-245",                    - DHCP Option 245          - custom defined
"option-246",                    - DHCP Option 246          - custom defined
"option-247",                    - DHCP Option 247          - custom defined
"option-248",                    - DHCP Option 248          - custom defined
"option-249",                    - DHCP Option 249          - custom defined
"option-250",                    - DHCP Option 250          - custom defined
"option-251",                    - DHCP Option 251          - custom defined
"option-252",                    - DHCP Option 252          - custom defined
"option-253",                    - DHCP Option 253          - custom defined
"option-254",                    - DHCP Option 254          - custom defined
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $option->name();
#Modify name
$option->name("host-name");
```

num()

Use this method to set or retrieve the numeric value of the DHCP option.

Note that specifying "num" without "name" is supported only for predefined options and custom options with names that haven't been modified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the number of the DHCP option. It must be an integer ranging from 1 to 254.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get num
my $num = $option->num();
#Modify num
$option->num("12");
```

value( )

Use this method to set or retrieve the custom value of the DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the data type of the DHCP option. Valid values are

```
  "boolean"                          - Specify "true" or "false" for boolean values.

  "ip-address"                       - Specify an IPv4 address. An IPv4 address is a 32-bit number
in dotted decimal notation.
                                       It consists of four 8-bit groups of decimal digits separated
                                       by decimal points (example: "192.168.1.2").

  "array of ip-address"              - Specify a list of IPv4 addresses, each separated by a comma.
                                       (example: "192.168.1.2, 10.11.12.13, 79.2.3.1")

  "string"                           - Specify a value in ASCII text string or
                                       a list of hexadecimal characters (0-9, a-f, or A-F) without
spaces and separated by colons.

  "text"                             - Specify a value in ASCII text string.

  "8-bit unsigned integer"           - Specify a numeric value between 0 and 255.

  "array of 8-bit unsigned integer"   - Specify a list of 8-bit unsigned integers, each separated by
a comma.

  "16-bit unsigned integer"          - Specify a numeric value between 0 and 65535.

  "array of 16-bit unsigned integer"  - Specify a list of 16-bit unsigned integers, each separated
by a comma.

  "32-bit unsigned integer"          - Specify a numeric value between 0 and 4294967295.

  "array of 32-bit unsigned integer"  - Specify a list of 32-bit unsigned integers, each separated
by a comma.

  "8-bit signed integer"             - Specify a numeric value between -128 and 127.

  "array of 8-bit signed integer"     - Specify a list of 8-bit signed integers, each separated by a
comma.

  "16-bit signed integer"            - Specify a numeric value between -32768 and 32767.

  "array of 16-bit signed integer"    - Specify a list of 16-bit signed integers, each separated by
a comma.

  "32-bit signed integer"            - Specify a numeric value between -2147483648 and 2147483647.

  "array of 32-bit signed integer"    - Specify a list of 32-bit signed integers, each separated by
a comma.

  "domain-list"                      - Specify a list of domain names. Add a space between each
name.
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get value
my $value = $option->value();
#Modify value
$option->value("abc.com");
```

vendor_class( )

Use this method to set an option space name as the vendor class value of a DHCP option or to retrieve an option space name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the defined option space name in string format. The default option space name is "DHCP".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get vendor_class
my $name = $option->vendor_class();
#Modify vendor_class
$option->vendor_class("DHCP");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP Option object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP Option object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP Option object

```
my $dhcp_option = Infoblox::DHCP::Option->new(
    name         => "subnet-mask",
    num          => "1",
    value        => "255.0.0.0",
    vendor_class => "DHCP"
);
unless($dhcp_option) {
       die("Construct DHCP option object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP option object created successfully\n";
```

#Get the Grid DHCP object and add a DHCP option to it

```
#Get the Grid DHCP object
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DHCP",
    grid   => "Infoblox"
);
my $object = $retrieved_objs[0];
```

```
unless ($object) {
       die("Get Grid DHCP object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get Grid DHCP object found at least 1 matching entry\n";

#Apply the changes to the Grid DHCP object
$object->options([$dhcp_option]);

#Apply the changes
$session->modify($object)
    or die("Modify Grid DHCP object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Grid DCHP object with DHCP option updated to Infoblox appliance successfully\n";
```

#Modify DHCP option

```
#Modify an existing option object

#Modifying the name of the DHCP option object.
$dhcp_option->name("host-name");

#Modifying the value of the DHCP option object.
$dhcp_option->value("infoblox.com");

#Modifying the number of the DHCP option object.
$dhcp_option->num("12");

#Apply changes to the Grid DHCP object.
$object->options([$dhcp_option]);

#Update Grid DHCP object through the Infoblox session.
$session->modify($object)
            or die("modify Grid DHCP object failed: ",
                $session->status_code() . ":" . $session->status_detail());
       print "Grid DHCP object with modified DHCP option updated to Infoblox appliance successfully\n";
```

#Remove DHCP option

```
#Apply changes to the Grid DHCP object.
$object->options([]);

#Update Grid DHCP object through the Infoblox session.
$session->modify($object)
            or die("modify Grid DHCP object failed: ",
                $session->status_code() . ":" . $session->status_detail());
       print "Removed DHCP option from Grid DHCP object successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify(),Infoblox::Grid::DHCP, Infoblox::Grid::Member::DHCP, Infoblox::DHCP::Network, Infoblox::DHCP::NetworkTemplate, Infoblox::DHCP::SharedNetwork, Infoblox::DHCP::FixedAddr, Infoblox::DHCP::FixedAddrTemplate, Infoblox::DHCP::Range, Infoblox::DHCP::RangeTemplate, Infoblox::DHCP::Filter::Option

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Option60MatchRule - DHCP Option60MatchRule object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DHCP->option60_match_rules( )
  - Infoblox::Grid::Member::DHCP->option60_match_rules( )
- METHODS
  - match_value( )
  - option_space( )
  - is_substring( )
  - substring_offset( )
  - substring_length( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Option60MatchRule - DHCP Option60MatchRule object.

DESCRIPTION

An Option60MatchRule object allows specifying match rules for Option 60 globally.

CONSTRUCTOR

```
my $option60_match_rule = Infoblox::DHCP::Option60MatchRule->new(
                            match_value      => $string,          #Required
                            option_space     => $string,          #Required
                            is_substring     => "true" | "false", #Optional / Default is undefined
                            substring_offset => $num,             #Optional / Default is undefined
                            substring_length => $num              #Optional / Default is undefined
                          );
```

MODULE METHODS

The following functions are available to be applied to a DHCP Option60MatchRule object.

Infoblox::Grid::DHCP->option60_match_rules( )

Use this function to specify match rule for option60 at grid level. See Infoblox::Grid::DHCP->option60_match_rules() for parameters and return values.

Example

```
#Construct a DHCP Option60MatchRule object
my $option60_match_rule = Infoblox::DHCP::Option60MatchRule->new(
                                        "option_space" => "infospace",
                                        "match_value" => "Nortel-i2004-A",
                                        "is_substring" => "true",
                                        "substring_offset" => "8",
                                        "substring_length" => "14"
                                      );

# Configure DHCP Option60MatchRule on the Grid DHCP object
my $response = $grid_dhcp->option60_match_rules([$option60_match_rule]);
```

Infoblox::Grid::Member::DHCP->option60_match_rules( )

Use this function to specify match rule for option60 at member level. See Infoblox::Grid::Member::DHCP->option60_match_rules() for parameters and return values.

Example

```
#Construct a DHCP Option60MatchRule object
my $option60_match_rule = Infoblox::DHCP::Option60MatchRule->new(
```

```
                                        "option_space" => "infospace",
                                        "match_value" => "Nortel-i2004-A",
                                        "is_substring" => "true",
                                        "substring_offset" => "10",
                                        "substring_length" => "24"
                                      );

# Configure DHCP Option60MatchRule on the Grid Member DHCP object
my $response = $grid_member_dhcp->option60_match_rules([$option60_match_rule]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP Option60MatchRule object.

match_value( )

Use this method to set or retrieve the match value of a DHCP Option60MatchRule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired match value in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get match_value
my $match_value = $option60_match_rule->match_value();
# Modify match_value
$option60_match_rule->match_value("SUNW");
```

option_space( )

Use this method to set or retrieve the option space of a DHCP Option60MatchRule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an string that contains defined Infoblox::DHCP::OptionSpace object. See Infoblox::DHCP::OptionSpace for parameters and return values.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get option_space
my $match_value = $option60_match_rule->option_space();
# Modify option_space
$option60_match_rule->option_space("new_option");
```

is_substring( )

Use this method to set or retrieve the is_substring flag of a DHCP Option60MatchRule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the is_substring flag or "false" to deactivate/unset it. The default value for this field is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get is_substring
my $is_substring = $option60_match_rule->is_substring();
# Modify is_substring
$option60_match_rule->is_substring("true");
```

substring_offset( )

Use this method to set or retrieve the substring_offset of a DHCP Option60MatchRule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The substring_offset for the DHCP Option60MatchRule object (range from 0 to 65535). The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get substring_offset
my $substring_offset = $option60_match_rule->substring_offset();
# Modify substring_offset
$option60_match_rule->substring_offset("10");
```

substring_length( )

Use this method to set or retrieve the substring_length of a DHCP Option60MatchRule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The substring_length for the DHCP Option60MatchRule object (range from 0 to 65535). The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get substring_length
my $substring_length = $option60_match_rule->substring_length();
# Modify substring_length
$option60_match_rule->substring_length("23");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP Option60MatchRule insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

```
my $option_space1 = Infoblox::DHCP::OptionSpace->new(
                                                "name" => "infospace"
                                                );
my $response = $session->add( $option_space1);
print "Add option space operation success \n" if ($response );
print "Add option space operation failed  \n" if ( not $response );

#Create a DHCP OptionDefinition object.>
my $option60_match_rule = Infoblox::DHCP::Option60MatchRule->new(
                                                "option_space" => "infospace",
                                                "match_value" => "Nortel-i2004-A",
                                                "is_substring" => "true",
                                                "substring_offset" => "0",
                                                "substring_length" => "14"
                                                );
unless ($option60_match_rule) {
     die("Construct DHCP Option60MatchRule failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

#Get on Grid DHCP object to add Option60MatchRule object>
my @result = $session->get(
    object => "Infoblox::Grid::DHCP",
    grid   => "Infoblox"
    )
    or die("get DHCP failed: ",
               $session->status_code(). ":" .$session->status_detail());
print "get grid DHCP Objcet successful.\n";

unless (scalar(@result) == 0) {
my $griddhcp = $result[0];
if ($griddhcp) {

 $griddhcp->option60_match_rules([$option60_match_rule]);

  #Applying the changes to appliance through session.
  $session->modify($griddhcp)
      or die("Modify DHCP failed: ",
         $session->status_code(). ":" .$session->status_detail());
  print"Modified Grid DHCP Object successfully.\n";
    }
    } else {
      print "No Grid DHCP object found.";
   }

#To remove Option60MatchRule object in Grid DHCP object>
my @result = $session->get(
    object => "Infoblox::Grid::DHCP",
    grid   => "Infoblox"
    )
    or die("get DHCP failed: ",
               $session->status_code(). ":" .$session->status_detail());
print "get grid DHCP Objcet successful.\n";

unless (scalar(@result) == 0) {
my $griddhcp = $result[0];
if ($griddhcp) {

$griddhcp->option60_match_rules(undef);

  #Applying the changes to appliance through session.
  $session->modify($griddhcp)
      or die("Modify DHCP failed: ",
         $session->status_code(). ":" .$session->status_detail());
  print"Modified Grid DHCP Object successfully to remove Option60MatchRule object.\n";
  }
    } else {
      print "No Grid DHCP object found.";
   }

#Get OptionSpace object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::OptionSpace",
    name   => "infospace"
);
my $object = $retrieved_objs[0];
unless  $object
```

```
        (        ) {
    die("Get OptionSpace object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get OptionSpace object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove OptionSpace object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "OptionSpace object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Grid::DHCP, Infoblox::Grid::Member::DHCP, Infoblox::DHCP::OptionSpace, Infoblox::Session->get(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::OptionDefinition - DHCP option definition object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->search( )
  - Infoblox::Session->remove( )
- METHODS
  - name( )
  - code( )
  - space( )
  - type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::OptionDefinition - DHCP option definition object

DESCRIPTION

An OptionDefinition defines a DHCP option within a specific option space. A custom option can be defined in the predefined DHCP option space or in the user-defined vendor option space. To define an option, add the option definition to the required option space.

CONSTRUCTOR

```
my $option_definition1 = Infoblox::DHCP::OptionDefinition->new(
                              name  => $string, #Required
                              code  => $num,    #Optional / Default is 1
                              space => $string, #Optional / Default is DHCP
                              type  => $string, #Optional / Default is string
                             );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that can be applied to a DHCP option definition object.

Infoblox::Session->add( )

Use this method to add a option definition object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Create a DHCP OptionDefinition object.
my $option_definition1 = Infoblox::DHCP::OptionDefinition->new(
                                 space => "SUNW",
                                 name  => "JumpStart-server",
                                 code  => "6",
                                 type  => "text"
                                 );
# Submit for adding
my $response = $session->add( $option_definition1 );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching DHCP option definition objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP option definition object:
```

```
  name - A name for the option definition in string format (regular expression).
  code - Option number. Either a name or code is required.
  space- Optional. The option space in which the option is defined. The default is the DHCP option
space.
```

Example

```
#Getting option by name
my  @result_array = $session->get(
                    object   => "Infoblox::DHCP::OptionDefinition",
                    space    => "SUNW",
                    name     => "JumpStart-server",
                    );

#Getting option by code
my @result_array = $session->get(
                    object   => "Infoblox::DHCP::OptionDefinition",
                    space    => "DHCP",
                    code     => 176,
                    );
```

Infoblox::Session->modify( )

Use this method to modify a DHCP option definition object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify code
$object->code(20);
# Submit modification
my $response = $session->modify($object);
```

Infoblox::Session->search( )

Use this method to retrieve all the matching DHCP option definition objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP option definition object:

  name - A name for the option definition in string format (regular expression).
  code - Option number. Either a name or code is required.
  space- Optional. The option space in which the option is defined. The default is the DHCP option
space.
```

Example

```
#Getting option by name
my  @result_array = $session->search(
                    object   => "Infoblox::DHCP::OptionDefinition",
                    space    => "SUNW",
                    name     => "JumpStart-server",
                    );

#Getting option by code
my @result_array = $session->search(
                    object   => "Infoblox::DHCP::OptionDefinition",
                    space    => "DHCP",
                    code     => 176,
                    );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP option definition object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` to retrieve the specific object, and then submit this object for removal.

Example

```
my  @result_array = $session->get(
                    object   => "Infoblox::DHCP::OptionDefinition",
                    space    => "SUNW",
                    name     => "JumpStart-server",
                    );
```

```
# find the desired object from retrieved list.
my $desired_obj = $result_array[0];

# Submit for removal
my $response = $session->remove($desired_obj);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP option definition object.

name( )

Use this method to set or retrieve the name of a DHCP option definition object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $option_definitionname = $option_definition1 ->name();
# Modify name
$option_definition1 ->name("Start-server");
```

code( )

Use this method to set or retrieve the code of a DHCP option definition object. An option code number is used to identify the DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number for the option code. Value must be between 1 and 254. Values 1-79, 81-82, 85-87, 89, 91-92, 98, 112-114, 118-119, 124-125 are predefined. These option codes have corresponding predefined Types. Therefore, you cannot change the predefined Types.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get code
my $option_definitionname = $option_definition1 ->code();
# Modify code
$option_definition1 ->code(14);
```

space( )

Use this method to set or retrieve the space of a DHCP option definition object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the space name in string format.

Returns

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get space
my $option_space = $option_definition1 ->space();
```

```
# Modify space
$option_definition1 ->space("SUNW");
```

type( )

Use this method to set or retrieve the data type of the grid DHCP option.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The accepted types are the following:

```
  "boolean"                 - For "true" or "false" boolean values.

  "ip-address"              - For a single IPv4 address. An IPv4 address is a 32-bit number in dotted
decimal notation.
                              It consists of four 8-bit groups of decimal digits separated
                              by decimal points (example: "192.168.1.2").

  "array of ip-address"     - For a list of IPv4 addresses, each separated by a comma.
                              (example: "192.168.1.2, 10.11.12.13, 79.2.3.1")

  "string"                  - For value in ASCII text string or
                              a list of hexadecimal characters (0-9, a-f, or A-F) without spaces and
separated by colons.

  "text"                    - For value in ASCII text string.

  "domain-name"             - For domain name in ASCII text format

  "domain-list"             - For a list of domain names, each separated by a comma.

  "8-bit unsigned integer"  - For numeric value between 0 and 255.

  "16-bit unsigned integer" - For numeric value between 0 and 65535.

  "32-bit unsigned integer" - For numeric value between 0 and 4294967295.

  "8-bit signed integer"    - For numeric value between -128 and 127.

  "16-bit signed integer"   - For numeric value between -32768 and 32767.

  "32-bit signed integer"   - For numeric value between -2147483648 and 2147483647.

  "array of 8-bit unsigned integer"  - For a list of numeric values between 0 and 255, each
separated by a comma.

  "array of 16-bit unsigned integer" - For a list of numeric values between 0 and 65535, each
separated by a comma.

  "array of 32-bit unsigned integer" - For a list of numeric values between 0 and 4294967295, each
separated by a comma.

  "array of 8-bit integer"           - For a list of numeric values between -128 and 127, each
separated by a comma.

  "array of 16-bit integer"          - For a list of numeric values between -32768 and 32767, each
separated by a comma.

  "array of 32-bit integer"          - For a list of numeric values between -2147483648 and
2147483647, each separated by a comma.
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get type
my $option_definitiontype = $option_definition1 ->type();
# Modify type
$option_definition1 ->type("string");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP Network insertion

```
#PROGRAM STARTS: Include all the modules that will be used
```

```
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $option_space1 = Infoblox::DHCP::OptionSpace->new(
                                                "name" => "SUNW"
                                                );
my $response = $session->add( $option_space1);
print "Add option space operation success \n" if ($response );
print "Add option space operation failed  \n" if ( not $response );
```

#Create a DHCP OptionDefinition object.

```
my $option_definition1 = Infoblox::DHCP::OptionDefinition->new(
                                space => "SUNW",
                                name  => "JumpStart-server",
                                code  => "6",
                                type  => "text"
                              );
        unless ($option_definition1) {
        die("Construct DHCP OptionDefinition failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
  }
 print "DHCP OptionDefinition object created successfully\n";

 #Add the DHCP OptionDefinition object to Infoblox appliance through a session
       my $response = $session->add( $option_definition1 )
       or die("Add a DHCP  OptionDefinition object failed: ",
             $session->status_code() . ":" . $session->status_detail());

  print "Add a DHCP  OptionDefinition object successfully\n";
```

#Get and modify a DHCP OptionDefinition object attribute

```
#Get Network through the session
my  @retrieved_objs = $session->get(
               object   => "Infoblox::DHCP::OptionDefinition",
               space  => "SUNW",
               name   => "JumpStart-server",
            );

my $object = $retrieved_objs[0];

unless ($object) {
    die("Get DHCP OptionDefinition object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP OptionDefinition object found at least 1 matching entry\n";

#Modify one of the attributes of the specified Network attribute

 $object->code(20);    #Apply the changes

$session->modify($object)
    or die("Modify DHCP OptionDefinition attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "DHCP OptionDefinition object modified successfully \n";
```

#Remove a DHCP OptionDefinition Object

```
 #Get DHCP OptionDefinition object through the session
 my  @retrieved_objs = $session->get(
                  object =>"Infoblox::DHCP::OptionDefinition",
                  space  => "SUNW",
```

```
                    name   => "JumpStart-server",
            );

    my $object = $retrieved_objs[0];

    unless ($object) {
     die("Get DHCP OptionDefinition object failed: ",
         $session->status_code() . ":" . $session->status_detail());
     }

 print "Get DHCP OptionDefinition object found at least 1 matching entry\n";

#Submit the object for removal

 $session->remove($object)
     or die("Remove Network failed: ",
         $session->status_code() . ":" . $session->status_detail());

 print "DHCP OptionDefinition removed successfully \n";

 #Remove option space.

 my $response = $session->remove( object => "Infoblox::DHCP::OptionSpace" ,
                                  name   => "SUNW",
                                );
 print "Remove option space success \n" if ($response );
 print "Remove option space failed  \n" if ( not $response );

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::OptionSpace, Infoblox::DHCP::Option, Infoblox::Session->get(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::OptionSpace - DHCP OptionSpace object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - comment( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::OptionSpace - DHCP OptionSpace object.

DESCRIPTION

An Option Space defines a namespace in which vendor options can be defined. To define a specific vendor option space, add an option space to DHCP.

CONSTRUCTOR

```
my $option_space = Infoblox::DHCP::OptionSpace->new(
      name      => $string, #Required
      comment   => $string, #Optional, default is undefined
);
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that can be applied to a DHCP OptionSpace object.

Infoblox::Session->add( )

Use this method to add a DHCP OptionSpace object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a DHCP OptionSpace object
my $option_space = Infoblox::DHCP::OptionSpace->new(
      name => "infospace",
      comment => "some comment"
);
#Submit for addition
my $response = $session->add($option_space)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP OptionSpace object:

 name     - Required. Option space name in string format.
 comment  - Optional. Option space comment.
```

Example

```
#Get DHCP OptionSpace object through the session
my @retrieved_objs = $session->get(
```

```
    object => "Infoblox::DHCP::OptionSpace",
    name   => "infospace",
);
```

Infoblox::Session->modify( )

Use this method to modify a DHCP OptionSpace object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the name.
$option_space->name("info_space");
#Submit modification
my $response = $session->modify( $option_space );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP OptionSpace object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` to retrieve the specific DHCP OptionSpace object, and then submit this object for removal.

Example

```
#Get DHCP OptionSpace object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::OptionSpace",
    name   => "info_space",
);
#Find the desired object from the retrieved list.
my $desired_option_space = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_option_space );
```

Infoblox::Session->search( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP OptionSpace object:

 name     - Optional. Option space name in string format.
 comment  - Optional. Option space comment (regular expression).
```

Example

```
#Get DHCP OptionSpace object through the session
my @retrieved_objs = $session->search(
    object => "Infoblox::DHCP::OptionSpace",
    comment => ".*comm.*"
);
```

METHODS

This section describes all the methods that can used to set and retrieve the attribute values of a DHCP OptionSpace object.

name( )

Use this method to set or retrieve name of option space object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name in string format with a maximum of 256 bytes.

Returns

If parameter is specified, the method returns true when the modification succeeds, and returns false when the operation fails.

If parameter is not specified, the method returns the attribute value.

Example

```
#Get name
my $name = $option_space->name();
#Modify name
$option_space->name("info_space");
```

comment( )

Use this method to set or retrieve a comment about the option space object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If the parameter is specified, the method returns true when the modification succeeds, and returns false when the operation fails.

If the parameter is not specified, the method returns the attribute value.

Example

```
#Get comment
my $comment = $option_space->comment();
#Modify comment
$option_space->comment("my comment");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an OptionSpace object, such as get, add, modify and remove. Also sample includes error handling for the operations.

#Preparation prior to a DHCP OptionSpace object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create an OptionSpace object

```
my $option_space = Infoblox::DHCP::OptionSpace->new (
       "name" => "infospace"
);

unless($option_space) {
     die("Construct option_space filter failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "OptionSpace object created successfully\n";

#Verify if the DHCP OptionSpace exists in the Infoblox appliance
my $object = $session->get(
       object => "Infoblox::DHCP::OptionSpace",
       name   => "infospace"
);
unless ($object) {
   print "option space does not exist on server, safe to add the option_space\n";
   $session->add($option_space)
      or die("Add option space failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "DHCP OptionSpace added successfully\n";
```

#Get and Modify an OptionSpace object

```
#Get OptionSpace object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::OptionSpace",
    name   => "infospace"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get OptionSpace object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get OptionSpace object found at least 1 matching entry\n";

#Modify name of the obtained OptionSpace object
$object->name("info_space");

#Apply the changes
$session->modify($object)
    or die("Modify OptionSpace object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "OptionSpace object modified successfully \n";
```

#Remove an OptionSpace object

```
#Get OptionSpace object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::OptionSpace",
    name   => "info_space"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get OptionSpace object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get OptionSpace object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove OptionSpace object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "OptionSpace object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::OptionDefinition, Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::OrderedRanges - Ordered Ranges object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - network( )
  - ranges( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::OrderedRanges - Ordered Ranges object.

DESCRIPTION

An object type to set or retrieve an ordered list of DHCP ranges in a network.

CONSTRUCTOR

```
my $ordered_ranges = Infoblox::DHCP::OrderedRanges->new(
                     network                  => $network,                    #Required
                     ranges                   => [$range1, $range2, ...],     #Required
);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to an Ordered Ranges object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

No object is added in this method, but the ranges for the specified network assume the specified order.

Example

```
#Construct an object
my $ordered_ranges = Infoblox::DHCP::OrderedRanges->new(

network => $network,
                                                        ranges  => [$range1, $range2]
                                                       );
#Submit for addition
my $response = $session->add( $ordered_ranges );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific Ordered Ranges object:

network - Required. An Infoblox::DHCP::Network object.
```

Examples

```
my @retrieved_objs = $session->get(
    object   => "Infoblox::DHCP::OrderedRanges",
    network  => $network
```

```
);
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

No object is modified in this method, but the ranges for the specified network assume the specified order.

Example

```
#Use this method to modify the comment.
$ordered_ranges->ranges([$range4, $range2, $range3, $range1]);
#Submit modification
my $response = $session->modify( $ordered_ranges );
```

METHODS

Use the following methods to access the attributes of an Infoblox::DHCP::OrderedRanges object. Specify a parameter to set the attribute's value, or omit the parameter to get the attribute's value.

network( )

Use this method to set or retrieve the parent network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An Infoblox::DHCP::Network object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting network
my $network = $ordered_ranges->network();
#Modifying network
$ordered_ranges->network($nw1);
```

ranges( )

Use this method to set or retrieve an ordered list of DHCP ranges object references.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A list of Infoblox::DHCP::Range objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting ranges
my $ranges_ref = $ordered_ranges->ranges();
#Modifying ranges
$ordered_ranges->ranges([$range1, $range2, @range3]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to the range reordering

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;
```

```
#Create a session to the Infoblox appliance
my $host_ip =  "192.168.1.2";
my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
    );
unless($session){
        die("Constructor for session failed: ",
               Infoblox::status_code(). ":" . Infoblox::status_detail());
}
print "Session created successfully.\n";

my $network = Infoblox::DHCP::Network->new(
                                            network   => "10.0.0.0/255.255.0.0",
                                            comment   => "add network",
                                           );

$session->add($network)
    or die("Add network failed: ",
               $session->status_code(). ":" .$session->status_detail());

print"Network added successfully.\n";

my $range = Infoblox::DHCP::Range->new(
                                        start_addr              => "10.0.0.1",
                                        end_addr                => "10.0.0.10",
                                        network                 => "10.0.0.0/16",
                                        comment                 => "range 1",
                                       );

$session->add($range)
    or die("Add range failed: ",
               $session->status_code(). ":" .$session->status_detail());

print"Range added successfully.\n";

$range = Infoblox::DHCP::Range->new(
                                      start_addr              => "10.0.0.21",
                                      end_addr                => "10.0.0.30",
                                      network                 => "10.0.0.0/16",
                                      comment                 => "range 2",
                                    );

$session->add($range)
    or die("Add range failed: ",
               $session->status_code(). ":" .$session->status_detail());

print"Range added successfully.\n";

$range = Infoblox::DHCP::Range->new(
                                      start_addr              => "10.0.0.31",
                                      end_addr                => "10.0.0.40",
                                      network                 => "10.0.0.0/16",
                                      comment                 => "range 3",
                                    );

$session->add($range)
    or die("Add range failed: ",
               $session->status_code(). ":" .$session->status_detail());

print"Range added successfully.\n";

#Get the current range order.

my $tnetwork = Infoblox::DHCP::Network->new(
                                             network   => "10.0.0.0/255.255.0.0",
                                            );
my $or = $session->get(
     object  => "Infoblox::DHCP::OrderedRanges",
     network => $tnetwork
);

unless($or){
       die("Get ordered ranges failed: ",
                $session->status_code() . ":" . $session->status_detail());
```

```
        }

 print "Get ordered ranges object found at least 1 matching entry\n";

 my ($r1, $r2, $r3) = @{$or->ranges()};

 print "The current order of the ranges is : " . $r1->comment() . "," .
               $r2->comment() . "," . $r3->comment() . "\n";

#Modify the range order.

 $or->ranges([$r3, $r2, $r1]);

 #Apply the change
 $session->modify($or)
       or die("Modify ordered ranges failed: ",
               $session->status_code(). ":" .$session->status_detail());

 print "Ordered ranges modified successfully.\n";

 $or = $session->get(
     object  => "Infoblox::DHCP::OrderedRanges",
     network => $tnetwork
 );

 unless($or){
       die("Get ordered ranges failed: ",
               $session->status_code() . ":" . $session->status_detail());
       }

 print "Get ordered ranges object found at least 1 matching entry\n";

 ($r1, $r2, $r3) = @{$or->ranges()};

 print "The current order of the ranges is : " . $r1->comment() . "," .
               $r2->comment() . "," . $r3->comment() . "\n";

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com

SEE ALSO

Infoblox::DHCP::Network, Infoblox::DHCP::Range, Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Range - DHCP Range object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - authority( )
  - always_update_dns( )
  - bootfile( )
  - bootserver( )
  - comment( )
  - ddns_domainname( )
  - ddns_generate_hostname( )
  - deny_all_clients( )
  - deny_bootp( )
  - disable( )
  - email_list( )
  - enable_ddns( )
  - enable_dhcp_thresholds( )
  - enable_email_warnings( )
  - enable_known_clients_option( )
  - enable_snmp_warnings( )
  - enable_unknown_clients_option( )
  - end_addr( )
  - exclude( )
  - extensible_attributes( )
  - failover_assoc( )
  - filters( )
  - known_clients_option( )
  - lease_scavenge_time( )
  - name( )
  - next_available_ip( )
  - ignore_dhcp_option_list_request( )
  - member( )
  - ms_options( )
  - network( )
  - network_view( )
  - nextserver( )
  - options( )
  - override_ddns_domainname( )
  - override_lease_scavenge_time( )
  - override_update_dns_on_lease_renewal( )
  - pxe_lease_time( )
  - range_high_water_mark( )
  - range_low_water_mark( )
  - recycle_leases( )
  - start_addr( )
  - template( )
  - unknown_clients_option( )
  - update_dns_on_lease_renewal( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Range - DHCP Range object.

DESCRIPTION

A DHCP range defines the specified range of IP addresses in a network. A DHCP range should be added for a network so the Infoblox appliance can assign IP addresses within that specified range to DHCP clients. If the client is on a network that is assigned a DHCP range, the device distributes an available IP address from that range to the DHCP client, or to a DHCP relay agent if the request came

through an agent. The DHCP range should also be assigned with a device. If devices are in a grid, the particular member serving DHCP for the DHCP range must be specified. If the server is an independent device, this device must be specified as the member that serves the DHCP range.

CONSTRUCTOR

```
my $dhcp_range = Infoblox::DHCP::Range->new(

    end_addr                        => $ipv4addr,                                   #Required
    network                         => $network,                                    #Optional
    network_view                    => $NetworkView,                                #Optional / Default is
system-defined default network view
    start_addr                      => $ipv4addr,                                   #Required
    authority                       => "true" | "false",                            #Optional / Default is
undefined
    always_update_dns               => "true" | "false",                            #Optional / Default is
"false"
    bootfile                        => $filename | undef,                           #Optional / Default is
undefined
    bootserver                      => $ipv4addr | $fqdn | undef,                   #Optional / Default is
undefined
    comment                         => $string,                                     #Optional / Default is
undefined
    ddns_domainname                 => $string | undef,                             #Optional / Default is
undefined
    ddns_generate_hostname          => "true" | "false",                            #Optional / Default is
undefined
    deny_all_clients                => "true" | "false",                            #Optional / Default is
"false"
    deny_bootp                      => "true" | "false" | undef,                    #Optional / Default is
undefined
    disable                         => "true" | "false",                            #Optional / Default is
"false"
    email_list                      => [ "admin\@test.com",...] | undef,            #Optional / Default is
undefined
    enable_ddns                     => "true" | "false",                            #Optional / Default is
undefined
    enable_dhcp_thresholds          => "true" | "false",                            #Optional / Default is
"false"
    enable_email_warnings           => "true" | "false",                            #Optional / Default is
"false"
    enable_known_clients_option     => "true" | "false",                            #Optional / Default is
"false"
    enable_snmp_warnings            => "true" | "false",                            #Optional / Default is
"false"
    enable_unknown_clients_option   => "true" | "false",                            #Optional / Default is
"false"
    exclude                         => [$ExclusionRange1,$ExclusionRange2,...],     #Optional / Default is
empty
    extensible_attributes           => { $string => $string | $num, $string => [ $string | $num, ... ],
... } #Optional / Default is undefined
    failover_assoc                  => $string,                                     #Optional / Default is
undefined
    filters                         => [$MAC, $RelayAgent, $Option...],             #Optional / Default is
empty
    ignore_dhcp_option_list_request => "true" | "false" | undef,                    #Optional / Default is
undefined
    known_clients_option            => "Allow" | "Deny",                            #Optional / Default is
undefined
    lease_scavenge_time             => $number,                                     #Optional / Default is -1
    member                          => $Member,                                     #Optional / Default is
undefined
    ms_options                      => [$MSOption1,$MSOption2,...],                 #Optional / Default is
empty
    name                            => $string,                                     #Optional / Default is
undefined
    nextserver                      => $ipv4addr | $fqdn | undef,                   #Optional / Default is
undefined
    options                         => [$Option1,$Option2,...],                     #Optional / Default is
empty
    override_ddns_domainname        => "true" | "false",                            #Optional / Default
value is "false"
    override_lease_scavenge_time    => "true" | "false",                            #Optional / Default
value is "false"
    override_update_dns_on_lease_renewal  => "true" | "false",                      #Optional / Default
value is "false"
    pxe_lease_time                  => $num,                                        #Optional / Default is
undefined
    range_high_water_mark           => $num,                                        #Optional / Default is 0
    range_low_water_mark            => $num,                                        #Optional / Default is 95
    recycle_leases                  => "true" | "false",                            #Optional / Default is
```

```
undefined
     template                          => $string,                                    #Optional / Default is
undefined
     unknown_clients_option            => "Allow" | "Deny",                           #Optional / Default is
undefined
     update_dns_on_lease_renewal       => "true" | "false",                           #Optional / Default
value is "false"
      );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a DHCP range object.

Infoblox::Session->add( )

Use this method to add a DHCP range object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr                => "10.0.0.10",
        network                 => "10.0.0.0/8",
        start_addr              => "10.0.0.1",
        authority               => "true",
        bootfile                => "bootfile1.com",
        bootserver              => "someserver.com",
        comment                 => "add range",
        ddns_generate_hostname  => "true",
        deny_all_clients        => "true",
        deny_bootp              => "true",
        disable                 => "true",
        enable_ddns             => "true",
        exclude                 => [$exclusion_range],
        filters                 => [$filter],
        member                  => $memberdhcp,
        network                 => $nview,
        nextserver              => "3.3.3.3",
        options                 => [$option1],
        pxe_lease_time          => "3600",
        recycle_leases          => "true",
);

# Submit for addition
my $response = $session->add( $dhcp_range );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP range object:

start_addr              - Optional. Starting IP address of DHCP range within the network.
network_view            - Optional. Network view in which the DHCP range is located. By default, all
network views are searched.
extensible_attributes   - Optional. A hash reference containing extensible attributes.
failover_assoc          - Optional. The name of the associated failover peer.
member                  - Optional. The member or Microsoft server serving the range. The valid value
is an Infoblox::DHCP::Member or Infoblox::DHCP::MSServer object.
```

Examples

```
my @retrieved_objs = $session->get(
                     object     => "Infoblox::DHCP::Range",
                     start_addr => "10.0.0.1" );

my @retrieved_objs = $session->get(
                     object       => "Infoblox::DHCP::Range",
                     start_addr   => "10.0.0.1",
                     network_view => "default" );

my @retrieved_objs = $session->get(
                     object                => "Infoblox::DHCP::Range",
                     extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a DHCP range object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$dhcp_range->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $dhcp_range );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP range object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP range object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::Range",
    start_addr   => "10.0.0.1",
    newtork_view => "default" );
# Find the desired object from the retrieved list.
my $desired_dhcp_range = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_dhcp_range );
```

Infoblox::Session->search( )

Use this method to search for DHCP range objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP range object:

  network      - Optional. Network in CIDR format (regular expression).
  network_view - Optional. Network view in which the DHCP range is located. By default, all network
views are searched.
  start_addr   - Optional. IPv4 address in dotted decimal notation (regular expression).
  end_addr     - Optional. IPv4 address in dotted decimal notation (regular expression).
  comment      - Optional. Comment in string format (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
  failover_assoc        - Optional. The name of the associated failover peer.
  member                - Optional. The member or Microsoft server serving the range. The valid value
is an Infoblox::DHCP::Member or Infoblox::DHCP::MSServer object.
```

Examples

```
# search for DHCP range object with start address having last octet as 1
my @retrieved_objs = $session->search(
    object     => "Infoblox::DHCP::Range",
    network    => "10.0.0.0/.",
    network    => "default",
    start_addr => ".*1",
    comment    => ".*comment" );

# search for DHCP range objects defining "Santa Clara" for "Site" extensible attribute
my @retrieved_objs = $session->search(
    object               => "Infoblox::DHCP::Range",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DHCP range.

authority( )

Use this method to set or retrieve the authority flag of a DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the authority flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authority
my $authority = $dhcp_range->authority();
#Modify authority
$dhcp_range->authority("false");
```

always_update_dns( )

Use this method to set or retrieve the always_update_dns flag of a DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This flag only applies if the range is associated with a Microsoft (r) server. If it is set to "true", the DHCP server will always update DNS, regardless of the requests from the DHCP clients. If this flag is set to false, clients will be allowed to update DNS when they request it.

Parameter

Specify "true" to set the always_update_dns flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get always_update_dns
my $always_update_dns = $dhcp_range->always_update_dns();
#Modify always_update_dns
$dhcp_range->always_update_dns("false");
```

bootfile( )

Use this method to set or retrieve a bootfile name which is the name of the file the client must download.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::Range->bootserver() and Infoblox::DHCP::Range->nextserver().

Parameter

The boot file name in string format with a maximum of 128 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootfile
my $bootfile = $dhcp_range->bootfile();
#Modify bootfile
$dhcp_range->bootfile("bootfile2.com");
#Un-override bootfile
$dhcp_range->bootfile(undef);
```

bootserver( )

Use this method to set or retrieve a bootserver name which is the name of the server where the boot file is stored.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::Range->bootfile() and Infoblox::DHCP::Range->nextserver().

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootserver
my $bootserver = $dhcp_range->bootserver();
#Modify bootserver
$dhcp_range->bootserver("anotherserver.com");
#Un-override bootserver
$dhcp_range->bootserver(undef);
```

comment( )

Use this method to set or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $dhcp_range->comment();
#Modify comment
$dhcp_range->comment("Modifying the DHCP range comment");
```

ddns_domainname( )

Use this method to set or retrieve the ddns_domainname value of a DHCP Range object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_dddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

The dynamic DNS domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_domainname
my $ddns_domainname = $dhcp_range->ddns_domainname();
#Modify ddns_domainname
$dhcp_range->ddns_domainname("test_domain.com");
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname flag of a DHCP range.

Enable this method to allow only the DHCP server to generate a hostname.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ddns_generate_hostname flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_generate_hostname
my $ddns_generate_hostname = $dhcp_range->ddns_generate_hostname();
#Modify ddns_generate_hostname
$dhcp_range->ddns_generate_hostname("false");
```

deny_all_clients( )

Use this method to set or retrieve the deny_all_clients flag of a DHCP range.

Enable this method to deny all lease requests for the DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_all_clients flag or "false" to deactivate/unset it. The default value for this field is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_all_clients
my $deny_all_clients = $dhcp_range->deny_all_clients();
#Modify deny_all_clients
$dhcp_range->deny_all_clients("false");
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp flag of a DHCP range.

Enable this method to deny all BOOTP requests.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_bootp flag or "false" to deactivate/unset it. The default value is false which indicates that this attribute inherit the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $dhcp_range->deny_bootp();
#Modify deny_bootp
$dhcp_range->deny_bootp("false");
```

disable( )

Use this method to set or retrieve the disable flag of a DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value for this field is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $dhcp_range->disable();
#Modify disable
$dhcp_range->disable("false");
```

email_list( )

Use this method to set or retrieve the email list to which the appliance sends notifications when DHCP address usage for the grid crosses a threshold.

If you enter email addresses in the email_list, then the appliance sends DHCP threshold alarms to the email addresses.

Parameter

The valid value is an array reference that contains string objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get email_list
my @email_list = $range->email_list();
#Modify email_list
$range->email_list( [ "admin\@test.com" ] );
```

enable_ddns( )

Use this method to set or retrieve the enable_ddns flag of a DHCP range.

Enable this method to allow dynamic DNS updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_ddns flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ddns
my $enable_ddns = $dhcp_range->enable_ddns();
#Modify enable_ddns
$dhcp_range->enable_ddns("false");
```

enable_dhcp_thresholds( )

Use this method to set or retrieve DHCP thresholds at the DHCP address range level. These thresholds are the high and low watermarks. If DHCP range usage goes above or below these percentage values, the Infoblox appliance generates a syslog message and sends an SNMP trap, if configured.

Enable this flag to set DHCP threshold values.

Parameter

Specify "true" to set the enable_dhcp_thresholds flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcp_thresholds
my $enable_dhcp_thresholds = $range->enable_dhcp_thresholds();
#Modify enable_dhcp_thresholds
$range->enable_dhcp_thresholds("false");
```

enable_email_warnings( )

Use this method to enable the appliance to send email notifications when DHCP address usage crosses a high or low watermark threshold.

Enable this flag if you want the appliance to send DHCP threshold warnings to the "email_list" provided.

Parameter

Specify "true" to set the enable_email_warnings flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_email_warnings
my $enable_email_warnings = $range->enable_email_warnings();
#Modify enable_email_warnings
$range->enable_email_warnings("false");
```

enable_known_clients_option( )

Use this method to set or retrieve the enable_known_clients_option flag of a DHCP range.

You must set the enable_known_clients_option flag to true in order to use the known_clients_option attribute.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_known_clients_option flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_known_clients_option
my $enable_known_clients_option = $dhcp_range->enable_known_clients_option();
#Modify enable_known_clients_option
$dhcp_range->enable_known_clients_option("false");
```

enable_snmp_warnings( )

Use this method to enable the appliance to send an SNMP trap when DHCP address usage crosses a watermark threshold.

If the flag is enabled, then the appliance sends SNMP traps when DHCP address usage crosses a watermark threshold.

Parameter

Specify "true" to set the enable_snmp_warnings flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_snmp_warnings
my $enable_snmp_warnings = $range->enable_snmp_warnings();
#Modify enable_snmp_warnings
```

```
$range->enable_snmp_warnings("false");
```

enable_unknown_clients_option( )

Use this method to set or retrieve the enable_unknown_clients_option flag of a DHCP range.

You must set the enable_unknown_clients_option flag to true in order to use the unknown_clients_option attribute.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_unknown_clients_option flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_unknown_clients_option
my $enable_unknown_clients_option = $dhcp_range->enable_unknown_clients_option();
#Modify enable_unknown_clients_option
$dhcp_range->enable_unknown_clients_option("false");
```

end_addr( )

Use this method to set or retrieve the end address of the DHCP range.

An end address is the last IP address in the DHCP range available for the clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the end address of the DHCP range in IPv4 address format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get end_addr
my $end_addr = $dhcp_range->end_addr();
#Modify end_addr
$dhcp_range->end_addr("10.0.0.20");
```

exclude( )

Use this method to set or retrieve the exclusion range of IP addresses in the DHCP range which should not be assigned to clients.

If static addresses in the middle of an DHCP range are assigned to certain hosts, those addresses can be excluded from the DHCP range using this method.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::ExclusionRange objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get exclude
my $ref_exclude = $dhcp_range->exclude();
#Modify exclude
$dhcp_range->exclude([$exl1]); #$exl1 is an Infoblox::DHCP::ExclusionRange object
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP range object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $dhcp_range->extensible_attributes();
#Modify extensible attributes
$dhcp_range->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

failover_assoc( )

Use this method to set or retrieve the failover peer association name of the DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the failover peer association name in string format. The default value is undefined.

Use an empty string ('') to remove a failover peer association from the DHCP range. Using undef is deprecated and may be changed in the future.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get failover_assoc
my $failover_assoc = $dhcp_range->failover_assoc();
#Modify failover_assoc
$dhcp_range->failover_assoc("peer2");
#Remove failover_assoc (using undef for removal is deprecated)
$dhcp_range->failover_assoc("");
```

filters( )

Use this method to set or retrieve the filters of the DHCP range.

Filters are used to screen the address assignments if the addresses in the DHCP range needs to be assigned to specific hosts.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::FilterRule::MAC, Infoblox::DHCP::FilterRule::NAC, Infoblox::DHCP::FilterRule::RelayAgent, Infoblox::DHCP::FilterRule::Option objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get filters
my $ref_filters = $dhcp_range->filters();
```

```
#Modify filters
$dhcp_range->filters([$exl1]); #$exl1 is an Infoblox::DHCP::FilterRule::MAC,
Infoblox::DHCP::FilterRule::NAC, Infoblox::DHCP::FilterRule::RelayAgent, or
Infoblox::DHCP::FilterRule::Option object.
```

known_clients_option( )

Use this method to set or retrieve the known_clients_option option of a DHCP range.

Set this option to "Allow" or "Deny" to allow or prevent the allocation of an IP address from this pool to any client that has a host declaration.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "Allow" or "Deny" to set the known_clients_option flag value. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get known_clients_option
my $known_clients_option = $dhcp_range->known_clients_option();
#Modify known_clients_option
$dhcp_range->known_clients_option("Deny");
```

lease_scavenge_time( )

Use this method to set or retrieve the lease_scavenge_time value of a DHCP Range object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_lease_scavenge_time method to "true". Setting the parameter to undefined causes the appliance to use the default from the upper level and automatically resets the override_lease_scavenge_time attribute to "false".

Note that when lease_scavenge_time is set to a defined value and override_lease_scavenge_time is set to "false", the last operation takes precedence. Thus the sequence $object->lease_scavenge_time(604800); $object->override_lease_scavenge_time("false"); will set override_lease_scavenge_time to "false", and the sequence $object->override_lease_scavenge_time("false"); $object->lease_scavenge_time(604800); will result in override_lease_scavenge_time="true".

Parameter

An integer that specifies the period of time, in seconds, that free and backup leases remain in the database before they are automatically deleted. To disable lease scavenging, set the parameter to -1. The minimum positive value must be greater than 86400 seconds (1 day). The default value of this parameter is -1, which means that lease scavenging is disabled.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lease_scavenge_time
my $lease_scavenge = $dhcp_range->lease_scavenge_time();
#Modify lease_scavenge_time
$dhcp_range->lease_scavenge_time(30 * 24 * 60 * 60);
```

name( )

Use this method to set or retrieve a descriptive name for the range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The desired name in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $dhcp_range->name();
#Modify name
$dhcp_range->name("external");
```

next_available_ip( )

Use this method to get the next available IP address in the DHCP range object that was retrieved. Note that to use this method, the object MUST be retrieved via a get or search call, or have been created first with an add call.

Parameter

This method does not accept input parameters.

Returns

This method returns the next available IP address in the range in string format.

Within a DHCP Range object, the next available IP address is the first address in this range that matches all of the following rules:

```
   * There is no fixed address or host address record matching this IP address
   * There is no unmanaged record matching this IP address
   * The IP address is not contained in any excluded range of this DHCP range
   * There is no active lease matching this IP address (Deleted leases in the recycle bin are not
considered.)
```

Example

```
# Retrieve a particular range
my @retrieved_objs = $session->get(
                        object     => "Infoblox::DHCP::Range",
                        start_addr => "10.0.0.20"
);

$dhcp_range = $retrieved_objs[0];

# Print the next available IP address in the range
print $dhcp_range->next_available_ip() . "\n";
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the ignore_dhcp_option_list_request flag of a DHCP shared network object. If this flag is set to false all the defined DHCP options will be returned to the client. This setting overrides the upper-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate/unset it. The default value is undefined which indicates that this attribute inherit the upper-level setting..

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $dhcp_range->ignore_dhcp_option_list_request();
# Un-override ignore_dhcp_option_list_request
$dhcp_range->ignore_dhcp_option_list_request(undef);
# Modify ignore_dhcp_option_list_request
$dhcp_range->ignore_dhcp_option_list_request("true");
```

member( )

Use this method to set or retrieve the member or Microsoft server that serves the DHCP range.

The member or Microsoft server that serves DHCP for this IP address range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a defined Infoblox::DHCP::Member or Infoblox::DHCP::MSServer object. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get member
my $ref_member = $dhcp_range->member();
#Modify member
$dhcp_range->member($member1); #$member is an Infoblox::DHCP::Member object
```

ms_options( )

Use this method to set or retrieve the Microsoft server custom DHCP options that are associated with this DHCP range.

Note: You can set only options that are included in both the Microsoft server and NIOS predefined option definitions. Options that are not common to both are read only and cannot be modified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains defined Infoblox::DHCP::MSOption objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $ref_options = $dhcp_range->ms_options();
#Modify options
$dhcp_range->ms_options([$option12]); #$option12 is an Infoblox::DHCP::MSOption object
```

network( )

Use this method to set or retrieve the DHCP network of the address range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the address and netmask of network in CIDR format (example: 10.0.0.0/8).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network
my $network = $dhcp_range->network();
```

network_view( )

Use this method to set or retrieve the network view of the DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the DHCP range is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $dhcp_range->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$dhcp_range->network_view($nview);
```

nextserver( )

Use this method to set or retrieve the nextserver IP address which is the boot file server where the boot file is stored.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::Range->bootfile() and Infoblox::DHCP::Range->bootserver().

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nextserver
my $nextserver = $dhcp_range->nextserver();
#Modify nextserver
$dhcp_range->nextserver("4.4.4.4");
#Un-override nextserver
$dhcp_range->nextserver(undef);
```

options( )

Use this method to set or retrieve the options of the DHCP range.

DHCP options describe network configuration settings and various services available on the network. Setting the DHCP options for the DHCP range overrides the options set at the network level.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::Option objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $ref_options = $dhcp_range->options();
#Modify options
$dhcp_range->options([$option12]); #$option12 is an Infoblox::DHCP::Option object
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP range is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the

grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$dhcp_range->override_ddns_domainname( );
#Modifying override_ddns_domainname
$dhcp_range->override_ddns_domainname("true");
```

override_lease_scavenge_time( )

The override_lease_scavenge_time attribute controls whether the lease_scavenge_time value in the DHCP range is used, instead of the upper-level default.

The override_lease_scavenge_time attribute can be specified explicitly. It is also set implicitly when lease_scavenge_time is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for lease_scavenge_time. Set the parameter to "false" to inherit the setting for lease_scavenge_time from the upper level.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_lease_scavenge_time
my $override_lease_scavenge_time=$dhcp_range->override_lease_scavenge_time( );
#Modifying override_lease_scavenge_time
$dhcp_range->override_lease_scavenge_time("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the DHCP range is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$dhcp_range->override_update_dns_on_lease_renewal( );
#Modifying override_update_dns_on_lease_renewal
$dhcp_range->override_update_dns_on_lease_renewal("true");
```

pxe_lease_time( )

Use this method to set or retrieve PXE Lease Time value of the DHCP range.

The Preboot Execution Environment Lease Time value is used by some hosts to boot remotely from a server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get pxe_lease_time
my $pxe_lease_time = $dhcp_range->pxe_lease_time();
#Modify pxe_lease_time
$dhcp_range->pxe_lease_time("7200");
```

range_high_water_mark( )

Use this method to set or retrieve the range_high_water_mark value. If the percentage of allocated addresses exceeds this value, an SNMP trap is sent, unless this is a range served by a Microsoft server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A number that specifies the percentage of allocated addresses. The range is from 1 to 100. The high watermark value must be greater than the low watermark value. The default value is 95.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get range_high_water_mark
my $range_high_water_mark = $range->range_high_water_mark();
#Modify range_high_water_mark
$range->range_high_water_mark(80);
```

range_low_water_mark( )

Use this method to set or retrieve the range_low_water_mark value. If the percentage of allocated addresses drops below this watermark, an SNMP trap is sent, unless this is a range served by a Microsoft server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A number that specifies the percentage of allocated addresses. The range is from 1 to 100. The low watermark value must be lower than the high watermark value. The default low watermark number is 0.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get range_low_water_mark
my $range_low_water_mark = $range->range_low_water_mark();
#Modify range_low_water_mark
$range->range_low_water_mark(10);
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag of the DHCP range.

If the flag is enabled, the leases are kept in recycle bin until one week after expiration. When the flag is disabled, the leases are

irrecoverably deleted.

Parameter

Specify "true" to set the recycle_leases flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $dhcp_range->recycle_leases();
#Modify recycle_leases
$dhcp_range->recycle_leases("false");
```

start_addr( )

Use this method to set or retrieve the start address of the DHCP range.

A start address is the first IP address in the DHCP range available for the clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the start address of the DHCP range in IPv4 address format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get start_addr
my $start_addr = $dhcp_range->start_addr();
#Modify start_addr
$dhcp_range->start_addr("10.0.0.2");
```

template( )

Use this method to set or retrieve the DHCP range template name for a DHCP Range object.

Template is the name of a DHCP range template which will be used to create the DHCP range.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid name in sting format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

The template parameter can be used to only add a new DHCP range and cannot be used during object modify or returned during object get.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get template
my $template = $dhcp_range->template();
#Modify template
$dhcp_range->template("Template_Name");
```

unknown_clients_option( )

Use this method to set or retrieve the unknown_clients_option option of a DHCP range.

Set this option to "Allow" or "Deny" to allow or prevent the allocation of an IP address from this pool to any client that has no host declaration.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "Allow" or "Deny" to set the unknown_clients_option flag value. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get unknown_clients_option
my $unknown_clients_option = $dhcp_range->unknown_clients_option();
#Modify unknown_clients_option
$dhcp_range->unknown_clients_option("Deny");
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$dhcp_range->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$dhcp_range->update_dns_on_lease_renewal("true");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP range object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP Range object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

```
#Create a DHCP Member object
my $memberdhcp = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2"
);
unless($memberdhcp) {
       die("Construct member failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Member object created successfully\n";

#Create the Network object with the member
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/8",
    comment => "add network",
    members => [ $memberdhcp ]
);
unless($network) {
       die("Construct Network object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
    or die("Add Network object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Network object added to Infoblox appliance successfully\n";

#Create the Mac Filter object
my $mac_filter = Infoblox::DHCP::Filter::MAC->new(
    "name"    => "filter1",
    "comment" => "5 minutes"
);

unless($mac_filter) {
       die("Construct Mac Filter object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Mac Filter object created successfully\n";

#Add the Mac Filter object into the Infoblox appliance through a session
$session->add($mac_filter)
    or die("Add Mac Filter object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Mac Filter object added to Infoblox appliance successfully\n";

#Construct Custom Option object
my $option1 = Infoblox::DHCP::Option->new(
    type  => "1",
    value => "255.0.0.0", );

#Construct DHCP Mac Filter rule object
my $filter = Infoblox::DHCP::FilterRule::MAC->new(
    filter_name => "filter1",
    permission  => "grant"
);

#Construct exclusion range object
my $exclusion_range = Infoblox::DHCP::ExclusionRange->new(
    start_address => "10.0.0.5",
    end_address   => "10.0.0.7",
    comment       => "Excluded range"
);
```

#Create a DHCP Range object

```
my $dhcp_range = Infoblox::DHCP::Range->new(
        end_addr                => "10.0.0.10",
        network                 => "10.0.0.0/8",
        start_addr              => "10.0.0.1",
        authority               => "true",
        bootfile                => "bootfile1.com",
        bootserver              => "someserver.com",
        comment                 => "add range",
        ddns_generate_hostname  => "true",
        deny_all_clients        => "true",
```

```
        deny_bootp              => "true",
        disable                 => "true",
        enable_ddns             => "true",
        exclude                 => [$exclusion_range],
        filters                 => [$filter],
        member                  => $memberdhcp,
        nextserver              => "3.3.3.3",
        options                 => [$option1],
        pxe_lease_time          => "3600",
        recycle_leases          => "true",
);
unless($dhcp_range) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

#Add the DHCP Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to the network successfully\n";
```

#Create a DHCP Range object using a template

```
my $dhcp_range_template = Infoblox::DHCP::RangeTemplate->new(
        name                    => "custom_range_template",
        offset                  => "10",
        number_of_addresses     => "10",
);
unless($dhcp_range_template) {
       die("Construct DHCP Range template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range template object created successfully\n";

#Add the DHCP Range template object into the Infoblox appliance through a session
$session->add($dhcp_range_template)
    or die("Add DHCP Range template object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range template object added to the network successfully\n";

my $dhcp_range = Infoblox::DHCP::Range->new(
        start_addr              => "10.0.0.11",
        end_addr                => "10.0.0.15",
        network                 => "10.0.0.0/8",
        template                => "custom_range_template"
);
unless($dhcp_range) {
       die("Construct DHCP Range object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range object created successfully\n";

#Add the DHCP Range object into the Infoblox appliance through a session
$session->add($dhcp_range)
    or die("Add DHCP Range object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object added to the network successfully\n";
```

#Get and modify a DHCP Range

```
#Get a DHCP Range through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::Range",
    start_addr => "10.0.0.1",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP Range object found at least 1 matching entry\n";

#Modify one of the attributes of the specified DHCP Range
$object->comment ("modified comment");
```

```
#Un-override bootfile
$object->bootfile(undef);

#Un-override bootserver
$object->bootserver(undef);

#Un-override nextserver
$object->nextserver(undef);

#Apply the changes
$session->modify($object)
    or die("Modify DHCP Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object modified successfully \n";
```

#Search a DHCP Range

```
# search for DHCP range object with start address having last octet as 1
my @retrieved_objs = $session->search(
    object     => "Infoblox::DHCP::Range",
    network    => "10.0.0.0/.",
    start_addr => ".*1",
    comment    => ".*comment"
);

my $object = $retrieved_objs[0];
unless ($object) {
    die("Search DHCP range failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DHCP range object found at least 1 matching entry\n";
```

#Remove a DHCP Range

```
#Get a DHCP Range through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::Range",
    start_addr => "10.0.0.1",
    end_addr   => "10.0.0.10",
);

my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP Range object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove DHCP Range failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range object removed successfully \n";

#submit the Range Template object for removal
$session->remove($dhcp_range_template)
    or die("Remove DHCP Range Tempate failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range Templateobject removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::DHCP::ViewInfoblox::DHCP::ExclusionRange, Infoblox::DHCP::FilterRule::MAC, Infoblox::DHCP::FilterRule::NAC, Infoblox::DHCP::FilterRule::RelayAgent, Infoblox::DHCP::FilterRule::Option, Infoblox::DHCP::Member, Infoblox::DHCP::Option, the Infoblox::DHCP::RangeTemplate manpage

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::RangeTemplate - DHCP RangeTemplate object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::NetworkTemplate->range_templates( )
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
- METHODS
  - authority( )
  - bootfile( )
  - bootserver( )
  - comment( )
  - ddns_domainname( )
  - ddns_generate_hostname( )
  - deny_all_clients( )
  - deny_bootp( )
  - enable_ddns( )
  - enable_known_clients_option( )
  - enable_unknown_clients_option( )
  - exclude( )
  - extensible_attributes( )
  - failover_assoc( )
  - filters( )
  - known_clients_option( )
  - lease_scavenge_time( )
  - member( )
  - name( )
  - nextserver( )
  - number_of_addresses( )
  - offset( )
  - options( )
  - override_ddns_domainname( )
  - override_lease_scavenge_time( )
  - override_update_dns_on_lease_renewal( )
  - pxe_lease_time( )
  - recycle_leases( )
  - unknown_clients_option( )
  - update_dns_on_lease_renewal( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::RangeTemplate - DHCP RangeTemplate object.

DESCRIPTION

A DHCP range template works as a 'cookie cutter' and can be used to create real DHCP range object in a quick and consistent way. The DHCP range object created from a DHCP range template will inherit most properties defined in range template object. So, most of the DHCP range template properties are the same as the DHCP range object properties.

CONSTRUCTOR

```
my $dhcp_range_template = Infoblox::DHCP::RangeTemplate->new(

    name                    => $string,             #Required
    number_of_addresses      => $num,               #Required
    offset                  => $num,                #Required
    authority               => "true" | "false",     #Optional / Default is undefined
    bootfile                => $filename,           #Optional / Default is undefined
                               $ipv4addr   $fqdn    undef   #Optional / Default is undefined
```

```
    bootserver              =>          |       |       ,
    comment                 => $string,             #Optional / Default is undefined
    ddns_domainname         => $string | undef,     #Optional / Default is undefined
    ddns_generate_hostname  => "true" | "false",    #Optional / Default is undefined
    deny_all_clients        => "true" | "false",    #Optional / Default is "false"
    deny_bootp              => "true" | "false" | undef,    #Optional / Default is undefined
    enable_ddns             => "true" | "false",    #Optional / Default is undefined
    enable_known_clients_option   => "true" | "false",    #Optional / Default is "false"
    enable_unknown_clients_option => "true" | "false",    #Optional / Default is "false"
    exclude                 => [$ExclusionRange1,$ExclusionRange2,...],   #Optional / Default is empty
    extensible_attributes   => { $string => $string | $num, $string => [ $string | $num, ... ], ... }
#Optional / Default is undefined
    failover_assoc          => $string,             #Optional / Default is undefined
    filters                 => [$MAC,$RelayAgent,$Option,...],            #Optional / Default is undefined
    known_clients_option          => "Allow" | "Deny",    #Optional / Default is undefined
    lease_scavenge_time             => $number,           #Optional / Default is -1
    member                  => $Member,             #Optional / Default is undefined
    nextserver              => $ipv4addr | $fqdn | undef,           #Optional / Default is undefined
    options                 => [$Option1,$Option2,...],                   #Optional / Default is empty
    override_ddns_domainname  => "true" | "false",                     #Optional / Default value is
"false"
    override_lease_scavenge_time    => "true" | "false",                #Optional / Default value is
"false"
    override_update_dns_on_lease_renewal  => "true" | "false",          #Optional / Default value is
"false"
    pxe_lease_time          => $num,                #Optional / Default is undefined
    recycle_leases          => "true" | "false",    #Optional / Default is undefined
    unknown_clients_option        => "Allow" | "Deny",    #Optional / Default is undefined
    update_dns_on_lease_renewal  => "true" | "false",     #Optional / Default value is "false"
     );
```

MODULE METHODS

The following functions are available to be applied to a DHCP Range Template object.

Infoblox::DHCP::NetworkTemplate->range_templates( )

Use this function to specify a range template at network template on the Infoblox appliance. See Infoblox::DHCP::NetworkTemplate->range_templates() for parameters and return values.

Example

```
#Construct a range template object
my $dhcp_range_template = Infoblox::DHCP::RangeTemplate->new(
            name                    => "custom_range_template",
            number_of_addresses     => "10",
            offset                  => "10",
    );

my $template = Infoblox::DHCP::Template->new(
            name                    => "custom_range_template",
    );

#Configure range template in network template object
my $responce = $network_template->range_templates([$template]);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that can be applied to a DHCP range template object.

Infoblox::Session->add( )

Use this method to add a DHCP range template object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $dhcp_range_template = Infoblox::DHCP::RangeTemplate->new(
            name                    => "custom_range_template",
            number_of_addresses     => "10",
            offset                  => "10",
            authority               => "true",
            bootfile                => "bootfile1.com",
```

```
        bootserver              => "someserver.com",
        comment                 => "add range",
        ddns_generate_hostname  => "true",
        deny_all_clients        => "true",
        deny_bootp              => "true",
        enable_ddns             => "true",
        exclude                 => [$exclusion_range_template],
        filters                 => [$filter],
        member                  => $memberdhcp,
        nextserver              => "3.3.3.3",
        options                 => [$option1],
        pxe_lease_time          => "3600",
        recycle_leases          => "true",
);

# Submit for addition
my $response = $session->add( $dhcp_range_template );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP range template object:

name - Optional. The name of the DHCP range template.
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
my @retrieved_objs = $session->get(
                     object => "Infoblox::DHCP::RangeTemplate",
                     name   => "custom_range_template"
                     );

my @retrieved_objs = $session->get(
                     object                => "Infoblox::DHCP::RangeTemplate",
                     extensible_attributes => { 'Site' => 'Santa Clara' }
                    );
```

Infoblox::Session->search( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DHCP Range Template object:

name    - Optional. The name of the DHCP range template (regular expression).
comment - Optional. The comment of the DHCP range template (regular expression).
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
my @retrieved_objs = $session->search(
                     object  => "Infoblox::DHCP::RangeTemplate",
                     name    => "custom_range_template"
                     comment => ".*",
                   );

my @retrieved_objs = $session->search(
                     object                => "Infoblox::DHCP::RangeTemplate",
                     extensible_attributes => { 'Site' => 'Santa Clara' }
          );
```

Infoblox::Session->modify( )

Use this method to modify a DHCP range template object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$dhcp_range_template->comment("this is a modified comment");
```

```
# Submit modification
my $response = $session->modify( $dhcp_range_template );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP range template object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific DHCP range template object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::RangeTemplate",
    name   => "custom_range_template"
   );

# Find the desired object from the retrieved list.
my $desired_dhcp_range_template = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_dhcp_range_template );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP range template object.

authority( )

Use this method to set or retrieve the authority flag of a DHCP range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the authority flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authority
my $authority = $dhcp_range_template->authority();
#Modify authority
$dhcp_range_template->authority("false");
```

bootfile( )

Use this method to set or retrieve a bootfile name which is the name of the file the client must download.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::RangeTemplate->bootserver() and Infoblox::DHCP::RangeTemplate->nextserver().

Parameter

The boot file name in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootfile
my $bootfile = $dhcp_range_template->bootfile();
#Modify bootfile
$dhcp_range_template->bootfile("bootfile2.com");
#Un-override bootfile
$dhcp_range_template->bootfile(undef);
```

bootserver( )

Use this method to set or retrieve a bootserver name which is the name of the server where the boot file is stored.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::RangeTemplate->bootfile() and Infoblox::DHCP::RangeTemplate->nextserver().

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootserver
my $bootserver = $dhcp_range_template->bootserver();
#Modify bootserver
$dhcp_range_template->bootserver("anotherserver.com");
#Un-override bootserver
$dhcp_range_template->bootserver(undef);
```

comment( )

Use this method to set or retrieve a descriptive comment of a DHCP range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $dhcp_range_template->comment();
#Modify comment
$dhcp_range_template->comment("Modifying the DHCP range template comment");
```

ddns_domainname( )

Use this method to set or retrieve the ddns_domainname value of a DHCP Range template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_dddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

The dynamic DNS domain name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_domainname
my $ddns_domainname = $dhcp_range_template->ddns_domainname();
#Modify ddns_domainname
$dhcp_range_template->ddns_domainname("test_domain.com");
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname flag of a DHCP range template.

Enable this method to allow only the DHCP server to generate a hostname.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ddns_generate_hostname flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_generate_hostname
my $ddns_generate_hostname = $dhcp_range_template->ddns_generate_hostname();
#Modify ddns_generate_hostname
$dhcp_range_template->ddns_generate_hostname("false");
```

deny_all_clients( )

Use this method to set or retrieve the deny_all_clients flag of a DHCP range template.

Enable this method to deny all lease requests for the DHCP range created from template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_all_clients flag or "false" to deactivate/unset it. The default value for this field is false.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_all_clients
my $deny_all_clients = $dhcp_range_template->deny_all_clients();
#Modify deny_all_clients
$dhcp_range_template->deny_all_clients("false");
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp flag of a DHCP range template.

Enable this method to deny all BOOTP requests for DHCP range created from this template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_bootp flag or "false" to deactivate/unset it. The default value is false which indicates that this attribute inherit the upper-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $dhcp_range_template->deny_bootp();
```

```
#Modify deny_bootp
$dhcp_range_template->deny_bootp("false");
```

enable_ddns( )

Use this method to set or retrieve the enable_ddns flag of a DHCP range template.

Enable this method to allow dynamic DNS updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_ddns flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ddns
my $enable_ddns = $dhcp_range_template->enable_ddns();
#Modify enable_ddns
$dhcp_range_template->enable_ddns("false");
```

enable_known_clients_option( )

Use this method to set or retrieve the enable_known_clients_option flag of a DHCP range.

You must set the enable_known_clients_option flag to true in order to use the known_clients_option attribute.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_known_clients_option flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_known_clients_option
my $enable_known_clients_option = $dhcp_range->enable_known_clients_option();
#Modify enable_known_clients_option
$dhcp_range->enable_known_clients_option("false");
```

enable_unknown_clients_option( )

Use this method to set or retrieve the enable_unknown_clients_option flag of a DHCP range.

You must set the enable_unknown_clients_option flag to true in order to use the unknown_clients_option attribute.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_unknown_clients_option flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_unknown_clients_option
my $enable_unknown_clients_option = $dhcp_range->enable_unknown_clients_option();
#Modify enable_unknown_clients_option
$dhcp_range->enable_unknown_clients_option("false");
```

exclude( )

Use this method to set or retrieve the exclusion range template in the DHCP range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::ExclusionRangeTemplate objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get exclude
my $ref_exclude = $dhcp_range_template->exclude();
#Modify exclude
$dhcp_range_template->exclude([$exl1]); #$exl1 is an Infoblox::DHCP::ExclusionRangeTemplate
object
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP Range Template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $dhcp_range_template->extensible_attributes();
#Modify extensible attributes
$dhcp_range_template->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [
'Peter', 'Tom' ] });
```

failover_assoc( )

Use this method to set or retrieve the failover peer association name of the DHCP range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the failover peer association name in string format. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get failover_assoc
my $failover_assoc = $dhcp_range_template->failover_assoc();
#Modify failover_assoc
$dhcp_range_template->failover_assoc("peer2");
```

filters( )

Use this method to set or retrieve the filters of the DHCP range template.

Filters are used to screen the address assignments if the addresses in the DHCP range needs to be assigned to specific hosts.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains defined Infoblox::DHCP::FilterRule::MAC, Infoblox::DHCP::FilterRule::NAC, Infoblox::DHCP::FilterRule::RelayAgent, and Infoblox::DHCP::FilterRule::UserClass objects. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get filters
my $ref_filters = $dhcp_range_template->filters();
#Modify filters
$dhcp_range_template->filters([$filter1]); #$filter1 is an Infoblox::DHCP::FilterRule::MAC,
Infoblox::DHCP::FilterRule::MAC,  Infoblox::DHCP::FilterRule::RelayAgent, or
Infoblox::DHCP::FilterRule::UserClass object.
```

known_clients_option( )

Use this method to set or retrieve the known_clients_option option of a DHCP range.

Set this option to "Allow" or "Deny" to allow or prevent the allocation of an IP address from this pool to any client that has a host declaration.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "Allow" or "Deny" to set the known_clients_option flag value. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get known_clients_option
my $known_clients_option = $dhcp_range->known_clients_option();
#Modify known_clients_option
$dhcp_range->known_clients_option("Deny");
```

lease_scavenge_time( )

Use this method to set or retrieve the lease_scavenge_time value of a DHCP Range Template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_lease_scavenge_time method to "true". Setting the parameter to undefined causes the appliance to use the default from the upper level and automatically resets the override_lease_scavenge_time attribute to "false".

Note that when lease_scavenge_time is set to a defined value and override_lease_scavenge_time is set to "false", the last operation takes precedence. Thus the sequence $object->lease_scavenge_time(604800); $object->override_lease_scavenge_time("false"); will set override_lease_scavenge_time to "false", and the sequence $object->override_lease_scavenge_time("false"); $object->lease_scavenge_time(604800); will result in override_lease_scavenge_time="true".

Parameter

An integer that specifies the period of time, in seconds, that free and backup leases remain in the database before they are automatically deleted. To disable lease scavenging, set the parameter to -1. The minimum positive value must be greater than 86400 seconds (1 day). The default value of this parameter is -1, which means that lease scavenging is disabled.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lease_scavenge_time
my $lease_scavenge = $dhcp_range_template->lease_scavenge_time();
```

```
#Modify lease_scavenge_time
$dhcp_range_template->lease_scavenge_time(30 * 24 * 60 * 60);
```

member( )

Use this method to set or retrieve the member or Microsoft (r) server of the DHCP range template.

The member or Microsoft server that serves DHCP for this IP address range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a defined Infoblox::DHCP::Member or Infoblox::DHCP::MSServer object. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get member
my $ref_member = $dhcp_range_template->member();
#Modify member
$dhcp_range_template->member($member1); #$member is an Infoblox::DHCP::Member object
```

name( )

Use this method to set or retrieve the description of the name of the DHCP range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the fixed address in string format with a maximum of 64 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $dhcp_range_template->name();
#Modify name
$dhcp_range_template->name("custom_range_template");
```

nextserver( )

Use this method to set or retrieve the nextserver IP address which is the boot file server where the boot file is stored.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::RangeTemplate->bootfile() and Infoblox::DHCP::RangeTemplate->bootserver().

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nextserver
my $nextserver = $dhcp_range_template->nextserver();
#Modify nextserver
$dhcp_range_template->nextserver("4.4.4.4");
#Un-override nextserver
```

```
$dhcp_range_template->nextserver(undef);
```

number_of_addresses( )

Use this method to set or retrieve the number of addresses in this DHCP range template

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of addresses in this DHCP range template.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get number_of_addresses
my $number_of_addresses = $dhcp_range_template-> number_of_addresses();
#Modify number_of_addresses
$dhcp_range_template->number_of_addresses("10");
```

offset( )

Use this method to set or retrieve the address offset of this DHCP range template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The address offset of this DHCP range template. The offset is calculated from the start address of network template to which the DHCP range template is assigned to. For example, if offset is set to 10 and number_of_addresses is set to 100, and network is set to 10.0.0.0/8, then start address of DHCP range created from this range template will be 10.0.0.10 and end address will be 10.0.0.100.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get offset
my $offset = $dhcp_range_template->offset();
#Modify offset
$dhcp_range_template->offset("10");
```

options( )

Use this method to set or retrieve the options of the DHCP range template.

DHCP options describe network configuration settings and various services available on the network.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::Option objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $ref_options = $dhcp_range_template->options();
#Modify options
$dhcp_range_template->options([$option12]); #$option12 is an Infoblox::DHCP::Option object
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP range template is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$dhcp_range_template->override_ddns_domainname( );
#Modifying override_ddns_domainname
$dhcp_range_template->override_ddns_domainname("true");
```

override_lease_scavenge_time( )

The override_lease_scavenge_time attribute controls whether the lease_scavenge_time value in the DHCP range template is used, instead of the upper-level default.

The override_lease_scavenge_time attribute can be specified explicitly. It is also set implicitly when lease_scavenge_time is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for lease_scavenge_time. Set the parameter to "false" to inherit the setting for lease_scavenge_time from the upper level.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_lease_scavenge_time
my $override_lease_scavenge_time=$dhcp_range_template->override_lease_scavenge_time( );
#Modifying override_lease_scavenge_time
$dhcp_range_template->override_lease_scavenge_time("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the DHCP range template is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$dhcp_range_template->override_update_dns_on_lease_renewal( );
#Modifying override_update_dns_on_lease_renewal
$dhcp_range_template->override_update_dns_on_lease_renewal("true");
```

pxe_lease_time( )

Use this method to set or retrieve the PXE lease time value of the DHCP range template.

The Preboot Execution Environment lease time value is used by some hosts to boot remotely from a server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get pxe_lease_time
my $pxe_lease_time = $dhcp_range_template->pxe_lease_time();
#Modify pxe_lease_time
$dhcp_range_template->pxe_lease_time("7200");
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag of the DHCP range template.

If the flag is enabled, the leases are kept in recycle bin until one week after expiration. When the flag is disabled, the leases are irrecoverably deleted.

Parameter

Specify "true" to set the recycle_leases flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $dhcp_range_template->recycle_leases();
#Modify recycle_leases
$dhcp_range_template->recycle_leases("false");
```

unknown_clients_option( )

Use this method to set or retrieve the unknown_clients_option option of a DHCP range.

Set this option to "Allow" or "Deny" to allow or prevent the allocation of an IP address from this pool to any client that has no host declaration.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "Allow" or "Deny" to set the unknown_clients_option flag value. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get unknown_clients_option
my $unknown_clients_option = $dhcp_range->unknown_clients_option();
#Modify unknown_clients_option
$dhcp_range->unknown_clients_option("Deny");
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$dhcp_range_template->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$dhcp_range_template->update_dns_on_lease_renewal("true");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP range template object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP RangeTemplate object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $host_ip = "192.168.1.2";
my $session = Infoblox::Session->new(
               master   => $host_ip,      #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a DHCP Member object
my $memberdhcp = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => $host_ip
);
unless($memberdhcp) {
       die("Construct member failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Member object created successfully\n";
```

```
#Create the Mac Filter object
my $mac_filter = Infoblox::DHCP::Filter::MAC->new(
    name    => "filter1",
    comment => "5 minutes"
);

unless($mac_filter) {
       die("Construct Mac Filter object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Mac Filter object created successfully\n";

#Add the Mac Filter object into the Infoblox appliance through a session
$session->add($mac_filter)
    or die("Add Mac Filter object failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "Mac Filter object added to Infoblox appliance successfully\n";

#Construct DHCP Mac Filter rule object
my $filter = Infoblox::DHCP::FilterRule::MAC->new(
    filter_name => "filter1",
    permission  => "grant"
);
unless($filter){
       die("Construct filter object failed: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
 print "Filter object created successfully\n";

#Construct exclusion range template object
my $exclusion_range_template = Infoblox::DHCP::ExclusionRangeTemplate->new(
    offset              => "15",
    number_of_addresses => "5",
    comment             => "Excluded range"
);
unless($exclusion_range_template){
       die("Construct exclusion range template object failed: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
 print "Exclusion range template object created successfully\n";
```

#Create a DHCP RangeTemplate object

```
my $dhcp_range_template = Infoblox::DHCP::RangeTemplate->new(
        name                    => "custom_range_template",
        offset                  => "10",
        number_of_addresses     => "10",
        authority               => "true",
        bootfile                => "bootfile1.com",
        bootserver              => "someserver.com",
        comment                 => "add range",
        ddns_generate_hostname  => "true",
        deny_all_clients        => "true",
        deny_bootp              => "true",
        enable_ddns             => "true",
        exclude                 => [$exclusion_range_template],
        filters                 => [$filter],
        member                  => $memberdhcp,
        nextserver              => "3.3.3.3",
        pxe_lease_time          => "3600",
        recycle_leases          => "true",
);
unless($dhcp_range_template) {
       die("Construct DHCP RangeTemplate object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP RangeTemplate object created successfully\n";

#Add the DHCP RangeTemplate object into the Infoblox appliance through a session
$session->add($dhcp_range_template)
    or die("Add DHCP RangeTemplate object failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "DHCP RangeTemplate object added to the network successfully\n";
```

#Search a DHCP RangeTemplate

```
#Search a DHCP RangeTemplate through the session
my @retrieved_objs = $session->search(
    object     => "Infoblox::DHCP::RangeTemplate",
    name       => "custom_range_template",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Search DHCP RangeTemplate failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Search DHCP RangeTemplate object found at least 1 matching entry\n";
```

#Get and modify a DHCP RangeTemplate

```
#Get a DHCP RangeTemplate through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::RangeTemplate",
    name       => "custom_range_template",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP RangeTemplate failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP RangeTemplate object found at least 1 matching entry\n";

#Modify one of the attributes of the specified DHCP RangeTemplate
$object->comment("modified comment");

#Un-override bootfile
$object->bootfile(undef);

#Un-override bootserver
$object->bootserver(undef);

#Un-override nextserver
$object->nextserver(undef);

#Apply the changes
$session->modify($object)
    or die("Modify DHCP Range Template failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "DHCP RangeTemplate object modified successfully \n";
```

#Remove a DHCP RangeTemplate

```
#Get a DHCP RangeTemplate through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DHCP::RangeTemplate",
    name       => "custom_range_template",
);

my $object = $retrieved_objs[0];
unless ($object) {
       die("Get DHCP Range Template failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get DHCP RangeTemplate object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove DHCP Range Template failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "DHCP Range Template object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::NetworkTemplate, Infoblox::DHCP::ExclusionRangeTemplate, Infoblox::DHCP::Template, Infoblox::DHCP::FilterRule::MAC, Infoblox::DHCP::FilterRule::NAC, Infoblox::DHCP::FilterRule::RelayAgent, Infoblox::DHCP::FilterRule::UserClass, Infoblox::DHCP::Member, Infoblox::DHCP::Option

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::RoamingHost - DHCP Roaming Host object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - address_type( )
  - bootfile( )
  - bootserver( )
  - client_identifier_prepend_zero( )
  - comment( )
  - ddns_domainname( )
  - ddns_hostname( )
  - deny_bootp( )
  - dhcp_client_identifier( )
  - disable( )
  - discovered_name( )
  - discoverer( )
  - enable_ddns( )
  - extensible_attributes( )
  - first_discovered( )
  - force_roaming_hostname( )
  - ignore_dhcp_option_list_request( )
  - ipv6_client_hostname( )
  - ipv6_ddns_domainname( )
  - ipv6_ddns_hostname( )
  - ipv6_domain_name( )
  - ipv6_domain_name_servers( )
  - ipv6_duid( )
  - ipv6_enable_ddns( )
  - ipv6_force_roaming_hostname( )
  - ipv6_match_option( )
  - ipv6_options( )
  - ipv6_template( )
  - last_discovered( )
  - mac( )
  - match_client( )
  - name( )
  - netbios( )
  - network_component_description( )
  - network_component_ip( )
  - network_component_name( )
  - network_component_port_description( )
  - network_component_port_name( )
  - network_component_port_number( )
  - network_component_type( )
  - network_view( )
  - nextserver( )
  - options( )
  - os( )
  - override_ddns_domainname( )
  - override_ipv6_ddns_domainname( )
  - override_ipv6_domain_name( )
  - override_ipv6_domain_name_servers( )
  - override_ipv6_enable_ddns( )
  - override_ipv6_options( )
  - override_preferred_lifetime( )
  - override_valid_lifetime( )
  - port_duplex( )
  - port_link_status( )
  - port_speed( )
  - port_status( )
  - port_vlan_description( )
  - port_vlan_name( )
  - port_vlan_number( )
  - preferred_lifetime( )

- pxe_lease_time( )
- template( )
- valid_lifetime( )

- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::RoamingHost - DHCP Roaming Host object.

DESCRIPTION

A Roaming Host is a specific host that a DHCP server always assigns when a lease request comes from a particular MAC address of the client.

CONSTRUCTOR

```
my $roaming_host = Infoblox::DHCP::RoamingHost -> new(
      name                            => $string,                        #Required
      address_type                    => "ipv4" | "ipv6" | "both",       #Optional / Default is "ipv4"
      bootfile                        => $filename,                      #Optional / Default is empty
      bootserver                      => $ipv4addr,                      #Optional / Default is empty
      client_identifier_prepend_zero  => "true" | "false",               #Optional / Default is "false"
      comment                         => $string,                        #Optional / Default is empty
      ddns_domainname                 => $string,                        #Optional / Default is empty
      ddns_hostname                   => $string,                        #Optional / Default is empty
      deny_bootp                      => "true" | "false",               #Optional / Default is "false"
      dhcp_client_identifier          => $string,                        #Optional / Default is empty
      disable                         => "true" | "false",               #Optional / Default is "false"
      enable_ddns                     => "true" | "false",               #Optional / Default is "false"
      extensible_attributes           => { $string => $string | $num, $string => [ $string | $num, ... ],
... }, #Optional / Default is undefined
      force_roaming_hostname          => "true" | "false",               #Optional / Default is "false"
      ignore_dhcp_option_list_request => "true" | "false" | undef,       #Optional / Default is "false"
      ipv6_ddns_domainname            => $string,                        #Optional / Default is undefined
      ipv6_ddns_hostname              => $string,                        #Optional / Default is empty
      ipv6_domain_name_servers        => $string,                        #Optional / Default is undefined
      ipv6_domain_name                => $string,                        #Optional / Default is undefined
      ipv6_duid                       => $duid,                          #Optional / Default is undefined
      ipv6_enable_ddns                => "true" | "false" | undef,       #Optional / Default is undefined
      ipv6_force_roaming_hostname     => "true" | "false",               #Optional / Default is "false"
      ipv6_match_option               => "duid" | undef,                 #Optional / Default is undefined
      ipv6_options                    => [$option1, $option2,...],       #Optional / Default is undefined
      ipv6_template                   => $string,                        #Optional / Default is undefined
      last_discovered                 => $string,                        #Optional / Default is empty
      mac                             => $mac_addr,                      #Optional / Default is empty
      match_client                    => "MAC" | "CLIENT_IDENTIFIER",    #Optional / Default is "MAC"
      network_view                    => $NetworkView,                   #Optional / Default is "default"
network view
      nextserver                      => $ipv4addr,                      #Optional / Default is empty
      options                         => [$option1, $option2, ...],      #Optional / Default is undefined
      override_ddns_domainname        => "true" | "false",               #Optional / Default value is "false"
      override_ipv6_ddns_domainname   => "true" | "false",               #Optional / Default is "false"
      override_ipv6_domain_name_servers => "true" | "false",             #Optional / Default is "false"
      override_ipv6_domain_name       => "true" | "false",               #Optional / Default is "false"
      override_ipv6_enable_ddns       => "true" | "false",               #Optional / Default is "false"
      override_ipv6_options           => "true" | "false",               #Optional / Default is "false"
      override_preferred_lifetime     => "true" | "false",               #Optional / Default is "false"
      override_valid_lifetime         => "true" | "false",               #Optional / Default is "false"
      preferred_lifetime              => $num,                           #Optional / Default is 27000
      pxe_lease_time                  => $time,                          #Optional / Default is empty
      template                        => $string,                        #Optional / Default is undefined
      valid_lifetime                  => $num,                           #Optional / Default is 43200
);
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DHCP Roaming Host object.

Infoblox::Session->add( )

Use this method to add a DHCP Roaming Host object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct a DHCP Roaming Host object
my $roaming_host = Infoblox::DHCP::RoamingHost->new(
        "name"                              => "JohnSmith",
        "bootfile"                          => "bootfile",
        "bootserver"                        => "bootserver.com",
        "client_identifier_prepend_zero"    => "true",
        "comment"                           => "add roaming host using uid",
        "ddns_domainname"                   => "ddns.domain.com",
        "ddns_hostname"                     => "ddns.host.domain.com",
        "deny_bootp"                        => "false",
        "dhcp_client_identifier"            => "foo",
        "disable"                           => "false",
        "enable_ddns"                       => "false",
        "match_client"                      => "MAC",
        "network_view"                      => $nview,
        "nextserver"                        => "3.3.3.3",
        "options"                           => [ $option1,  $option2, $option3,  $option4, $option5,
$option6 ],
        "pxe_lease_time"                    => "3600",
);

#Submit for addition
my $response = $session->add($roaming_host)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

Apply the following attributes to get a specific DHCP Roaming Host object:

```
  name                    - Optional. The roaming host name.
  mac                     - Optional. MAC address of the roaming host.
  dhcp_client_identifier  - Optional, DHCP option 61 client identifier.
  network_view            - Optional. Network view in which the roaming host is located. By default,
all network views are searched.
  extensible_attributes   - Optional. A hash reference containing extensible attributes.
```

Examples

```
#Get DHCP Roaming Host object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::RoamingHost",
    mac    => "11:11:11:11:11.11",
);
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::RoamingHost",
    dhcp_client_identifier => "foo",
);
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::RoamingHost",
    name     => "JohnSmith",
    network_view => "default",
);

my @retrieved_objs = $session->get(
   object                => "Infoblox::DHCP::RoamingHost",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a DHCP Roaming Host object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the pxe_lease_time
$roaming_host->pxe_lease_time("1234");
```

```
#Submit modification
my $response = $session->modify( $roaming_host );
```

Infoblox::Session->remove( )

Use this method to remove a DHCP Roaming Host object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use get() or search() to retrieve the specific DHCP Roaming Host, and then submit this object for removal.

Example

```
#Get DHCP Roaming Host object through the session
my @retrieved_objs = $session->get(
    object       => "Infoblox::DHCP::RoamingHost",
    mac          => "11:11:11:11:11:11",
    network_view => "default"
);
#Find the desired object from the retrieved list.
my $desired_roaming_host = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_roaming_host );
```

Infoblox::Session->search( )

Use this method to search for DHCP Roaming Host object and matching Host object with MAC address in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

Apply the following attributes to search for a specific DHCP Roaming Host object:

```
  name           - Optional. The roaming host name (regular expression).
  mac            - Optional. MAC address of the roaming host (regular expression).
  network_view   - Optional. Network view in which the roaming host is located. By default, all
network views are searched.
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
# Search for all DHCP Roaming Host objects that are specified in the network "10.0.0.0/8" in the
"default" network view.
my @retrieved_objs = $session->search(
    object       => "Infoblox::DHCP::RoamingHost",
    mac          => "11:11:11:11:11.*",
    network_view => "default"
);

# Search for all DHCP Roaming Host objects with "Santa Clara" for the "Site" extensible attribute
my @retrieved_objs = $session->search(
    object                => "Infoblox::DHCP::RoamingHost",
    extensible_attributes => { 'Site' => 'Santa Clara' }
);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a DHCP Roaming Host object.

address_type( )

Use this method to set or retrieve the address_type field.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The possible values for address_type are the following: ipv4 - Roaming hosts are set up for IPv4 addresses only. ipv6 - Roaming hosts are set up for IPv6 addresses only. both - Both types of addresses are set up.

The default value of the parameter is "ipv4".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get address_type
my $address_type = $roaminghost->address_type();
#Modify address_type
$roaminghost->address_type("both");
```

bootfile( )

Use this method to set or retrieve the bootfile value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the file the client must download.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootfile
my $bootfile = $roaming_host->bootfile();
#Modify bootfile
$roaming_host->bootfile("boot_file");
```

bootserver( )

Use this method to set or retrieve the bootserver of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the server on which the boot file is stored.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootserver
my $bootserver = $roaming_host->bootserver();
#Modify bootserver
$roaming_host->bootserver("bootserver2.com");
```

client_identifier_prepend_zero( )

Use this method to prepend "\000" to the dhcp-client-identifier of a roaming host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

When you set the client_identifier_prepend_zero parameter to "true", it prefixes a 0 (zero) to the DHCP client identifier whenever option 61 is used to assign roaming hosts. The following is an excerpt from "man dhcp-options":

Please be aware that some DHCP clients, when configured with client identifiers that are ASCII text, will prepend a zero to the ASCII text. So you may need to write: option dhcp-client-identifier "\000foo"; rather than: option dhcp-client-identifier "foo";

Returns

If you specified a parameter, the method returns true when the modification succeeds and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get client_identifier_prepend_zero
my $client_identifier_prepend_zero = $roaming_host->client_identifier_prepend_zero();
#Modify dhcp_client_identifier
$roaming_host->client_identifier_prepend_zero("true");
```

comment( )

Use this method to set or retrieve the descriptive comment of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $roaming_host->comment();
#Modify comment
$roaming_host->comment("Modified DHCP Roaming Host object comment");
```

ddns_domainname( )

Use this method to set or retrieve the DDNS domain name value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_domainname attribute to "false".

Note that when ddns_domainname is set to a defined value and override_dddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_domainname("testdomain"); $object->override_ddns_domainname("false"); will set override_ddns_domainname to "false", and the sequence $object->override_ddns_domainname("false"); $object->ddns_domainname("testdomain"); will result in override_ddns_domainname="true".

Parameter

Specify the DDNS domain name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the DDNS domain name
my $ddns_domainname = $roaming_host->ddns_domainname();
#Modify the DDNS domain name
$roaming_host->ddns_domainname("ddns.domain.com");
```

ddns_hostname( )

Use this method to set or retrieve the DDNS host name value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the DDNS host name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the DDNS host name
my $ddns_hostname = $roaming_host->ddns_hostname();
#Modify the DDNS host name
$roaming_host->ddns_hostname("host.ddns.domain.com");
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp flag of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_bootp flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $roaming_host->deny_bootp();
#Modify deny_bootp
$roaming_host->deny_bootp("true");
```

dhcp_client_identifier( )

Use this method to set or retrieve the dhcp-client-identifier value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Use the dhcp_client_identifier or uid to assign roaming hosts if the DHCP clients send option 61. A DHCP client identifier can be a string or a MAC address. Set the "match_client" field to "CLIENT_IDENTIFIER" for roaming hosts to use the dhcp_client_identifier instead of the mac_address.

Returns

If you specified a parameter, the method returns true when the modification succeeds and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dhcp_client_identifier
my $uid = $roaming_host->dhcp_client_identifier();
#Modify dhcp_client_identifier
$roaming_host->dhcp_client_identifier("foo");
or
$roaming_host->dhcp_client_identifier("1:11::11::11::11::11::12");
```

disable( )

Use this method to set or retrieve the disable flag of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $roaming_host->disable();
#Modify disable
$roaming_host->disable("true");
```

discovered_name( )

Use this method to retrieve the discovered name of a DHCP Roaming Host object.

Parameter

none

Returns

The method returns the discovered name.

Example

```
#Get discovered_name
my $discovered_name = $roaming_host->discovered_name();
```

discoverer( )

Use this method to retrieve the discoverer of a DHCP Roaming Host object.

Parameter

none

Returns

The method returns the discoverer value.

Example

```
#Get discoverer
my $discoverer = $roaming_host->discoverer();
```

enable_ddns( )

Use this method to set or retrieve the enable_ddns flag of a DHCP Roaming Host object.

If enable_ddns value is specified as "true", then the A and PTR records corresponding to the Roaming Host object will be placed on the DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_ddns flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ddns
my $enable_ddns = $roaming_host->enable_ddns();
#Modify enable_ddns
$roaming_host->enable_ddns("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP Roaming Host object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $roaming_host->extensible_attributes();
#Modify extensible attributes
$roaming_host->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom'
```

```
] });
```

first_discovered( )

Use this method to retrieve the date and time that the DHCP Roaming Host object was first discovered.

Parameter

none

Returns

The method returns the date and time that the DHCP Roaming Host object was first discovered.

Example

```
#Get first_discovered
my $first_discovered = $roaming_host->first_discovered();
```

force_roaming_hostname( )

Use this method to set or retrieve the force_roaming_hostname flag of a DHCP Roaming Host object.

If the force_roaming_hostname value is "true", then the roaming host name is used as the ddns_hostname.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the force_roaming_hostname flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get force_roaming_hostname
my $force_roaming_hostname = $roaming_host->force_roaming_hostname();
#Modify force_roaming_hostname
$roaming_host->force_roaming_hostname("true");
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the ignore_dhcp_option_list_request flag of a DHCP roaming host object. If this flag is set to false all the defined DHCP options will be returned to the client.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate/unset it. The default value is false which indicates that this attribute inherit the member level setting..

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $roaming_host->ignore_dhcp_option_list_request();
# Un-override ignore_dhcp_option_list_request
$roaming_host->ignore_dhcp_option_list_request(undef);
# Modify ignore_dhcp_option_list_request
$roaming_host->ignore_dhcp_option_list_request("true");
```

ipv6_client_hostname( )

Use this method to retrieve the ipv6_client_hostname of a DHCP Roaming Host object. This method specifies the host name that the DHCP client sends to the Infoblox appliance using DHCP option 12. The method is read-only.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get ipv6_client_hostname
my $ipv6_client_hostname = $roaming_host->ipv6_client_hostname();
```

ipv6_ddns_domainname( )

Use this method to set or retrieve the IPv6 DDNS domain name value for the roaming host.

Setting this method to a defined value implicitly sets the override_ipv6_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_ipv6_ddns_domainname attribute to "false".

Note that when ipv6_ddns_domainname is set to a defined value and override_ipv6_ddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_ddns_domainname("testdomain"); $object->override_ipv6_ddns_domainname("false"); will set override_ipv6_ddns_domainname to "false", and the sequence $object->override_ipv6_ddns_domainname("false"); $object->ipv6_ddns_domainname("testdomain"); will result in override_ipv6_ddns_domainname="true".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DDNS domain name in FQDN (Fully Qualified Domain Name) format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_domainname
my $ipv6_ddns_domainname = $roaming_host->ipv6_ddns_domainname();
#Modify ipv6_ddns_domainname
$roaming_host->ipv6_ddns_domainname("domain.com");
```

ipv6_ddns_hostname( )

Use this method to set or retrieve the IPv6 DDNS host name value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the IPv6 DDNS host name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the IPv6 DDNS host name
my $ipv6_ddns_hostname = $roaming_host->ipv6_ddns_hostname();
#Modify the IPv6 DDNS host name
$roaming_host->ipv6_ddns_hostname("host.ddns.domain.com");
```

ipv6_domain_name( )

Use this method to set or retrieve the ipv6_domain_name value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_domain_name method to "true". Setting the parameter to undefined causes the appliance to use the upper level default and automatically resets the override_ipv6_domain_name attribute to "false".

Note that when ipv6_domain_name is set to a defined value and override_ipv6_domain_name is set to "false", the last operation takes

precedence. Thus the sequence $object->ipv6_domain_name("testdomain"); $object->override_ipv6_domain_name("false"); will set override_ipv6_domain_name to "false", and the sequence $object->override_ipv6_domain_name("false"); $object->ipv6_domain_name("testdomain"); will result in override_ipv6_domain_name="true".

Parameter

The name, in string format, of the client's domain for resolving names in DNS.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_domain_name
my $ipv6_domain_name = $roaming_host->ipv6_domain_name();
#Modify ipv6_domain_name
$roaming_host->ipv6_domain_name("test_domain.com");
```

ipv6_domain_name_servers( )

Use this method to set or retrieve the ipv6_domain_name_servers value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_domain_name_servers method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_ipv6_domain_name_servers attribute to "false".

Note that when ipv6_domain_name_servers is set to a defined value and override_ipv6_domain_name_servers is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_domain_name_servers("2001::1,2001::2"); $object->override_ipv6_domain_name_servers("false"); will set override_ipv6_domain_name_servers to "false", and the sequence $object->override_ipv6_domain_name_servers("false"); $object->ipv6_domain_name_servers("2001::1,2001::2"); will result in override_ipv6_domain_name_servers="true".

Parameter

The IPv6 addresses of the DNS servers.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_domain_name_servers
my $ipv6_domain_name_servers = $roaming_host->ipv6_domain_name_servers();
#Modify ipv6_domain_name_servers
$roaming_host->ipv6_domain_name_servers("2001::1,2001::2");
```

ipv6_duid( )

Use this method to set or retrieve the DHCPv6 Unique Identifier (DUID) of the Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DUID of the matching client in hexadecimal format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_duid
my $ipv6_duid = $roaming_host->ipv6_duid();
#Modify ipv6_duid
$roaming_host->ipv6_duid("1c2edfaa");
```

ipv6_enable_ddns( )

Use this method to set or retrieve the ipv6_enable_ddns flag for the roaming host. Set this flag to enable DHCPv6 dynamic DNS updates.

Setting this method to a defined value implicitly sets the override_ipv6_enable_ddns method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_ipv6_enable_ddns attribute to "false".

Note that when ipv6_enable_ddns is set to a defined value and override_ipv6_enable_ddns is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_enable_ddns("false"); $object->override_ipv6_enable_ddns("false"); will set override_ipv6_enable_ddns to "false", and the sequence $object->override_ipv6_enable_ddns("false"); $object->ipv6_enable_ddns("false"); will result in override_ipv6_enable_ddns="true".

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_enable_ddns
my $ipv6_enable_ddns = $roaming_host->ipv6_enable_ddns();
#Modify ipv6_enable_ddns
$roaming_host->ipv6_enable_ddns("true");
```

ipv6_force_roaming_hostname( )

Use this method to set or retrieve the ipv6_force_roaming_hostname flag of a DHCP Roaming Host object.

If the ipv6_force_roaming_hostname value is "true", then the roaming host name is used as the ddns_hostname. This flag applies to IPv6 roaming hosts.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ipv6_force_roaming_hostname flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_force_roaming_hostname
my $ipv6_force_roaming_hostname = $roaming_host->ipv6_force_roaming_hostname();
#Modify ipv6_force_roaming_hostname
$roaming_host->ipv6_force_roaming_hostname("true");
```

ipv6_match_option( )

Use this method to set or retrieve the ipv6_match_option value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

This field specifies the identification method for an IPv6 or mixed IPv4/IPv6 roaming host. Currently, the only supported value for this field is "duid", which corresponds to identification by DHCPv6 unique ID. The default value of this parameter is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_match_option
my $ipv6_match_option = $roaming_host->ipv6_match_option();
#Modify ipv6_match_option
$roaming_host->ipv6_match_option("duid");
```

ipv6_options( )

Use this method to set or retrieve the IPv6 options of a DHCP Roaming Host object.

Setting this method to a defined value implicitly sets the override_ipv6_options method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_ipv6_options attribute to "false".

Note that when ipv6_options is set to a defined value and override_ipv6_options is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_options([$option1]); $object->override_ipv6_options("false"); will set override_ipv6_options to "false", and the sequence $object->override_ipv6_options("false"); $object->ipv6_options([$option1]); will result in override_ipv6_options="true".

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_options
my $ipv6_options = $roaming_host->ipv6_options();
#Modify ipv6_options
my $option1 = Infoblox::DHCP::Option->new(
    name   => "dhcp6.info-refresh-time",
    value  => "1000"
);
$roaming_host->ipv6_options([$option1]);
```

ipv6_template( )

Use this method to set or retrieve the fixed address ipv6_template name for a DHCP Roaming Host object.

Specify the IPv6 fixed address template that is used to create the DHCP roaming host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The IPv6 fixed address template name in sting format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

The ipv6_template parameter can be used only to add a new DHCP roaming host. It cannot be used to modify an object or return an object during a get operation.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_template
my $ipv6_template = $roaming_host->ipv6_template();
#Modify ipv6_template
$roaming_host->ipv6_template("Template_Name");
```

last_discovered( )

Use this method to retrieve the time this object was last seen by a discovery job.

The default value for this field is empty.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get last_discovered
```

```
my $last_discovered = $roaming_host->last_discovered();
```

mac( )

Use this method to set or retrieve the MAC address value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

MAC Address of the DHCP roaming host object. Format for the hexadecimal strings for MAC addresses and vendor prefixes with colons or dashes. Both of the following formats are acceptable: 11:11:11:11:11:11 and 11-11-11-11-11-11.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mac
my $mac = $roaming_host->mac();
#Modify mac
$roaming_host->mac("11::11::11::11::11::12");
```

match_client( )

Use this method to set or retrieve the match_client value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

match_client "MAC": The fixed IP address is leased to the matching MAC address.
match_client "CLIENT_IDENTIFIER": The fixed IP address is leased to the matching DHCP client identifier
Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get match_client
my $match_client = $roaming_host->match_client();
#Modify match_client
$roaming_host->match_client("CLIENT_IDENTIFIER");
```

name( )

Use this method to set or retrieve the roaming host name associated with the DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

NOTE: To enable using roaming hosts, you must allow roaming hosts at the grid level by setting $griddhcp->enable_roaming_hosts("true");

Parameter

Desired roaming host name in string format with a maximum of 64 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get roaming host name
my $roaming_host_name = $roaming_host->name();
#Modify roaming host name
$roaming_host->name("JimmyJaz");
```

netbios( )

Use this method to retrieve the name in the NetBIOS reply that responded to a NetBIOS query.

Returns

The method returns the attribute value.

Example

```
#Get netbios
my $netbios = $roaming_host->netbios();
```

network_component_description( )

Use this method to retrieve the description of the network device that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the network device description.

Example

```
#Get network_component_description
my $network_component_description = $roaming_host->network_component_description();
```

network_component_ip( )

Use this method to retrieve the IP address of the network device that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the IP address of the network device.

Example

```
#Get network_component_ip
my $network_component_ip = $roaming_host->network_component_ip();
```

network_component_name( )

Use this method to retrieve the name of the network device that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the network device name.

Example

```
#Get network_component_name
my $network_component_name = $roaming_host->network_component_name();
```

network_component_port_description( )

Use this method to retrieve the description of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the description of the port on the network device.

Example

```
#Get network_component_port_description
my $network_component_port_description = $roaming_host->network_component_port_description();
```

network_component_port_name( )

Use this method to retrieve the name of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the name of the port on the network device.

Example

```
#Get network_component_port_name
my $network_component_port_name = $roaming_host->network_component_port_name();
```

network_component_port_number( )

Use this method to retrieve the number of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the network device port number in the range 0-9999.

Example

```
#Get network_component_port_number
my $network_component_port_number = $roaming_host->network_component_port_number();
```

network_component_type( )

Use this method to retrieve the type of the network device that is connected to DHCP Roaming Host object.

Parameter

none

Returns

The method returns the network device type.

Example

```
#Get network_component_type
my $network_component_type = $roaming_host->network_component_type();
```

network_view( )

Use this method to set or retrieve the network view of the DHCP roaming host

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the roaming host is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $roaming_host->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$roaming_host->network_view($nview);
```

nextserver( )

Use this method to set or retrieve the nextserver value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

IP address of the boot file server on which the boot file is stored.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nextserver
my $nextserver = $roaming_host->nextserver();
#Modify nextserver
$roaming_host->nextserver("3.3.3.4");
```

options( )

Use this method to set or retrieve the options of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

See Infoblox::DHCP::Option for parameters and methods available for this object.

Parameter

Options describe network configuration settings and various services available on the network. These options occur as variable-length fields at the end of DHCP messages.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $roaming_host->options();
#Modify options
$roaming_host->options([$option1, $option2]);
```

os( )

Use this method to retrieve the operating system associated with this roaming host.

Returns

The method returns the attribute value.

Example

```
#Get os
my $os = $roaming_host->os();
```

override_ddns_domainname( )

The override_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP Roaming Host is used, instead of the grid default.

The override_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_domainname
my $override_ddns_domainname=$roaming_host->override_ddns_domainname( );
#Modifying override_ddns_domainname
$roaming_host->override_ddns_domainname("true");
```

override_ipv6_ddns_domainname( )

The override_ipv6_ddns_domainname attribute controls whether the ddns_domainname value in the DHCP Roaming Host object is used, instead of the upper-level default.

The override_ipv6_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ipv6_ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for ddns_domainname. Set the parameter to "false" to inherit the upper-level setting for ipv6_ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_ddns_domainname
my $override_ipv6_ddns_domainname=$roaming_host->override_ipv6_ddns_domainname( );
#Modifying override_ipv6_ddns_domainname
$roaming_host->override_ipv6_ddns_domainname("true");
```

override_ipv6_domain_name( )

Use this method to set or retrieve the override_ipv6_domain_name value of a DHCP Roaming Host object.

The override_ipv6_domain_name attribute can be specified explicitly. It is also set implicitly when ipv6_domain_name is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for ipv6_domain_name. Set the parameter to "false" to inherit the upper-level setting for ipv6_domain_name.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_ipv6_domain_name
my $override_ipv6_domain_name = $roaming_host->override_ipv6_domain_name();
#Modify override_ipv6_domain_name
$roaming_host->override_ipv6_domain_name("test_domain.com");
```

override_ipv6_domain_name_servers( )

Use this method to set or retrieve the override_ipv6_domain_name_servers value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for ipv6_domain_name_servers. Set the parameter to "false" to inherit the upper-level setting for ipv6_domain_name_servers.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_ipv6_domain_name_servers
my $override_ipv6_domain_name_servers = $roaming_host->override_ipv6_domain_name_servers();
#Modify override_ipv6_domain_name_servers
$roaming_host->override_ipv6_domain_name_servers("true");
```

override_ipv6_enable_ddns( )

The override_ipv6_enable_ddns attribute controls whether the ipv6_enable_ddns value in the DHCP Roaming Host object is used, instead of the upper-level default.

The override_ipv6_enable_ddns attribute can be specified explicitly. It is also set implicitly when ipv6_enable_ddns is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for ipv6_enable_ddns. Set the parameter to "false" to inherit the upper-level setting for ipv6_enable_ddns.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_enable_ddns
my $override_ipv6_enable_ddns=$roaming_host->override_ipv6_enable_ddns( );
#Modifying override_ipv6_enable_ddns
$roaming_host->override_ipv6_enable_ddns("true");
```

override_ipv6_options( )

The override_ipv6_options attribute controls whether the ipv6_options value in the DHCP Roaming Host object is used, instead of the upper-level default.

The override_ipv6_options attribute can be specified explicitly. It is also set implicitly when ipv6_options is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for ipv6_options. Set the parameter to "false" to inherit the upper-level setting for ipv6_options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_options
my $override_ipv6_options=$roaming_host->override_ipv6_options( );
#Modifying override_ipv6_options
$roaming_host->override_ipv6_options("true");
```

override_preferred_lifetime( )

The override_preferred_lifetime attribute controls whether the preferred_lifetime value in the DHCP Roaming Host object is used, instead of the upper-level default.

The override_preferred_lifetime attribute can be specified explicitly. It is also set implicitly when preferred_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for preferred_lifetime. Set the parameter to "false" to inherit the upper-level setting for preferred_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_preferred_lifetime
my $override_preferred_lifetime=$roaming_host->override_preferred_lifetime( );
#Modifying override_preferred_lifetime
$roaming_host->override_preferred_lifetime("true");
```

override_valid_lifetime( )

The override_valid_lifetime attribute controls whether the valid_lifetime value in the DHCP Roaming Host object is used, instead of the upper-level default.

The override_valid_lifetime attribute can be specified explicitly. It is also set implicitly when valid_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for valid_lifetime. Set the parameter to "false" to inherit the upper-level setting for valid_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_valid_lifetime
my $override_valid_lifetime=$roaming_host->override_valid_lifetime( );
#Modifying override_valid_lifetime
$roaming_host->override_valid_lifetime("true");
```

port_duplex( )

Use this method to retrieve the duplex setting of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the port duplex setting. Returned values, if any, are one of the following:

Full
Half

Example

```
#Get port_duplex
my $port_duplex = $roaming_host->port_duplex();
```

port_link_status( )

Use this method to retrieve the link status of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the network device port link status. Returned values, if any, are one of the following: Connected Not Connected Unknown

Example

```
#Get port_link_status
my $port_link_status = $roaming_host->port_link_status();
```

port_speed( )

Use this method to retrieve the speed of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the network device port speed. Returned values, if any, are one of the following:

10 M
100 M
1G
10G
100G
Unknown

Example

```
#Get port_speed
my $port_speed = $roaming_host->port_speed();
```

port_status( )

Use this method to retrieve the status of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the network device port status. Returned values, if any, are one of the following:

Up
Down
Unknown

Example

```
#Get port_status
my $port_status = $roaming_host->port_status();
```

port_vlan_description( )

Use this method to retrieve the VLAN description of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the VLAN description of the network device port.

Example

```
#Get port_vlan_description
my $port_vlan_description = $roaming_host->port_vlan_description();
```

port_vlan_name( )

Use this method to retrieve the VLAN name of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the network device port VLAN name.

Example

```
#Get port_vlan_name
my $port_vlan_name = $roaming_host->port_vlan_name();
```

port_vlan_number( )

Use this method to retrieve the VLAN number of the network device port that is connected to the DHCP Roaming Host object.

Parameter

none

Returns

The method returns the port VLAN number in the range 0-9999.

Example

```
#Get port_vlan_number
my $port_vlan_number = $roaming_host->port_vlan_number();
```

preferred_lifetime( )

Use this method to set or retrieve the preferred lifetime of the roaming host.

Setting this method to a defined value implicitly sets the override_preferred_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_preferred_lifetime attribute to "false".

Note that when preferred_lifetime is set to a defined value and override_preferred_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->preferred_lifetime(20000); $object->override_preferred_lifetime("false"); will set override_preferred_lifetime to "false", and the sequence $object->override_preferred_lifetime("false"); $object->preferred_lifetime(20000); will result in override_preferred_lifetime="true".

Parameter

Preferred lifetime value in seconds. The default value is 27000. Note that the preferred lifetime cannot be greater than the valid lifetime.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get preferred_lifetime
my $preferred_lifetime = $roaming_host->preferred_lifetime();
#Modify preferred_lifetime
$roaming_host->preferred_lifetime(12345);
```

pxe_lease_time( )

Use this method to set or retrieve the pxe_lease_time value of a DHCP Roaming Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter appropriate values in the Days, Hours, Mins and Secs fields. Users can specify the duration of time it takes a host to connect to a boot server, such as a TFTP server, and download the file it needs to boot.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get pxe_lease_time
my $pxe_lease_time = $roaming_host->pxe_lease_time();
#Modify pxe_lease_time
$roaming_host->pxe_lease_time("369");
```

template( )

Use this method to set the fixed address template name for a DHCP Roaming Host object.

Template is the name of a DHCP Fixed Address template which will be used to create the DHCP Roaming Host.

Parameter

The valid name in string format.

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

The template parameter can be used to only add a new DHCP Roaming Host and cannot be used during object modify or returned during object get.

Example

```
#Add RoamingHost with template
my $roaming_host = Infoblox::DHCP::RoamingHost->new(
       "name"  => "JohnSmith",
       "template"  => "Template_Name",
);
#Submit for addition
my $response = $session->add($roaming_host)
```

valid_lifetime( )

Use this method to set or retrieve the preferred lifetime of the roaming host.

Setting this method to a defined value implicitly sets the override_valid_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_valid_lifetime attribute to "false".

Note that when valid_lifetime is set to a defined value and override_valid_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->valid_lifetime(20000); $object->override_valid_lifetime("false"); will set override_valid_lifetime to "false", and the sequence $object->override_valid_lifetime("false"); $object->valid_lifetime(20000); will result in override_valid_lifetime="true".

Parameter

Valid lifetime value in seconds. The default value is 43200.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get valid_lifetime
my $valid_lifetime = $roaming_host->valid_lifetime();
#Modify valid_lifetime
$roaming_host->valid_lifetime(12345);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

```
#Preparation prior to a DHCP Fixed Address object insertion

#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
```

```
my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Modify Grid DHCP object to enable support of the Roaming Hosts

```
my $grid_dhcp=$session->get(
                          object => "Infoblox::Grid::DHCP",
                          name=>"Infoblox"
                          );
unless ($grid_dhcp) {
    die("Get Grid DHCP object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Grid DHCP object found at least 1 matching entry\n";

my $save_grid_rh_status=$grid_dhcp->enable_roaming_hosts();
$grid_dhcp->enable_roaming_hosts("true");
unless($session->modify($grid_dhcp)) {
    die("Modifying Grid DHCP object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Roaming Hosts enabled at Grid level successfuly\n";

#Adding DHCP options
my $option1 = Infoblox::DHCP::Option->new(
                                          "name"  => "lease-time",
                                          "value" => "50"
                                         );

unless($option1) {
     die("Construct option1 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option1 object created successfully\n";

my $option2 = Infoblox::DHCP::Option->new(
                                          "type"      => "routers",
                                          "ipv4addrs" => ["10.0.0.1","10.0.0.255"],
                                         );

unless($option2) {
     die("Construct option2 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "option2 object created successfully\n";
```

#Create a DHCP Roaming Host object

```
#Create a DHCP Roaming Host object
my $roaming_host = Infoblox::DHCP::RoamingHost->new(
       "name"       => "JohnSmith",
       "mac"            => "00:00:00:11:11:11",
       "comment"        => "add roaming host",
       "enable_ddns"    => "true",
       "pxe_lease_time" => "3600",
       "bootserver"     => "someserver.com",
       "nextserver"     => "3.3.3.3",
       "bootfile"       => "bootfile.com",
       "options"        => [ $option1 , $option2 ],
       );
unless($roaming_host) {
     die("Construct roaming host failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Roaming Host object created successfully\n";
```

```
#Verify if the DHCP Roaming Host exists in the Infoblox appliance
my $object = $session->get(object => "Infoblox::DHCP::RoamingHost", mac => "00:00:00:11:11:11");
unless ($object) {
   print "roaming host does not exist on server, safe to add the roaming host\n";
   $session->add($roaming_host)
      or die("Add roaming host failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "DHCP Roaming Host added successfully\n";
```

#Search for a specific DHCP Roaming Host object

```
#Search for all DHCP Roaming Host objects with a given mac address
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::RoamingHost",
    mac     => "00:00:00:11:11.*",
     );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search for Roaming Host object failed: ",
              $session->status_code() . ":" . $session->status_detail());
}
print "Search Roaming Host object found at least 1 matching entry\n";
```

#Get and modify a DHCP Roaming Host object

```
#Get DHCP Roaming Host object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::RoamingHost",
    mac    => "00:00:00:11:11:11",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Roaming Host object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Roaming Host object found at least 1 matching entry\n";

#Modify one of the attributes of the obtained Roaming Host object
$object->pxe_lease_time("1234");

#Apply the changes
$session->modify($object)
    or die("Modify Roaming Host object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Roaming Host object modified successfully \n";
```

#Remove a Roaming Host object

```
#Get Roaming Host object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DHCP::RoamingHost",
    mac    => "00:00:00:11:11:11",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Roaming Host object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Roaming Host object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove Roaming Host object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Roaming Host object removed successfully \n";

#Restore Grid DHCP
$grid_dhcp->enable_roaming_hosts($save_grid_rh_status);
print "Grid DHCP setting restored successfully\n" if($session->modify($grid_dhcp));

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::SharedNetwork - DHCP Shared Network object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - authority( )
  - bootfile( )
  - bootserver( )
  - comment( )
  - ddns_generate_hostname( )
  - ddns_server_always_updates( )
  - ddns_update_fixed_addresses( )
  - ddns_ttl( )
  - ddns_use_option81( )
  - deny_bootp( )
  - disable( )
  - enable_ddns( )
  - extensible_attributes( )
  - ignore_dhcp_option_list_request( )
  - lease_scavenge_time( )
  - name( )
  - networks( )
  - network_view( )
  - nextserver( )
  - options( )
  - override_ddns_ttl( )
  - override_lease_scavenge_time( )
  - override_update_dns_on_lease_renewal( )
  - pxe_lease_time( )
  - update_dns_on_lease_renewal( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::SharedNetwork - DHCP Shared Network object

DESCRIPTION

You can create a shared network when two subnets share a particular network segment. Before creating a shared network, you must first create the subnetworks. For example, you must first create the networks 10.32.1.0 and 10.30.0.0 before designating them as a shared network.

CONSTRUCTOR

```
  my $sharednetwork = Infoblox::DHCP::SharedNetwork->new(
                  name                            => $string,                          #Required
                  networks                        => [ $Network1,$Network2,...],        #Required
                  authority                       => "true" | "false" | undef,          #Optional / Default
is undefined
                  bootfile                        => $filename | undef,                #Optional / Default
is undefined
                  bootserver                      => $ipv4addr | $fqdn | undef,
#Optional / Default is undefined
                  comment                         => $string | undef,                  #Optional / Default
is undefined
                  ddns_generate_hostname           => "true" | "false" | undef,         #Optional / Default
is undefined
                  ddns_server_always_updates       => "true" | "false",                 #Optional / Default
is "true"
                  ddns_update_fixed_addresses      => "true" | "false" | undef,         #Optional / Default
is undefined
```

```
                ddns_ttl                        => $num,                            #Optional / Default
is zero
                ddns_use_option81               => "true" | "false" | undef,        #Optional / Default
is undefined
                deny_bootp                      => "true" | "false" | undef,        #Optional / Default
is undefined
                disable                         => "true" | "false",                #Optional / Default
is "false"
                enable_ddns                     => "true" | "false" | undef,        #Optional / Default
is undefined
                extensible_attributes           => { $string => $string | $num, $string => [ $string |
$num, ... ], ... }, #Optional / Default is undefined
                ignore_dhcp_option_list_request => "true" | "false" | undef,        #Optional / Default
is "false"
                lease_scavenge_time             => $number,                         #Optional / Default
is -1
                network_view                    => $NetworkView,                    #Optional / Default
is "default" network view
                nextserver                      => $ipv4addr | $fqdn | undef,
#Optional / Default is undefined
                options                         => [$Option1,$Option2,...],         #Optional / Default
is empty list
                override_ddns_ttl               => "true" | "false",                #Optional / Default
value is "false"
                override_lease_scavenge_time    => "true" | "false",                #Optional / Default
value is "false"
                override_update_dns_on_lease_renewal => "true" | "false",           #Optional / Default
value is "false"
                pxe_lease_time                  => $num | undef,                    #Optional / Default
is undefined
                update_dns_on_lease_renewal     => "true" | "false",                #Optional / Default
value is "false"
              );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DHCP shared network object.

Infoblox::Session->add( )

Use this method to add a shared network object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object for network
my $network1 = Infoblox::DHCP::Network->new(
      network => "20.0.0.0/8",
);

#Construct an object shared network
  my $sharednetwork = Infoblox::DHCP::SharedNetwork->new(
      name     => "myname",
      networks => [$network1],
);

# Submit for adding sharednetwork
my $response = $session->add( $sharednetwork );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching shared network objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP shared network object:

 name          - Required.    The name of the shared network in string format.
 network_view  - Optional.    The network view in which the shared network is located. By default,
all network views are searched.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
 my  @result_array = $session->get(
      object       => "Infoblox::DHCP::SharedNetwork" ,
      name         => "myname",
      network_view => "default"
```

```
 );
```

Infoblox::Session->modify( )

Use this method to modify a shared network object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify enable_ddns method.
$object->enable_ddns("true");
# Submit modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove a shared network object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my  @result_array = $session->get(
      object       => "Infoblox::DHCP::SharedNetwork" ,
      name         => "myname",
      network_view => "default"
 );

# find the desired object from retrieved list.
my $desired_network = $result_array[0];

# Submit for removal
my $response = $session->remove( $desired_network );
```

Infoblox::Session->search( )

Use this method to search for DHCP shared network object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP shared network object:

 name          - Required. The name of the shared network in string format (regular expression).
 network_view  - Optional. The network view in which the shared network is located. By default, all
network views are searched.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DHCP shared network objects in the "default" network view
    my @retrieved_objs = $session->search(
      object    => "Infoblox::DHCP::SharedNetwork",
      name      => "my.*",
      network_view => "default"
    );

# search all Shared Networks Templates with a given extensible attribute 'Site'
    my @retrieved_objs = $session->search(
      object => "Infoblox::DHCP::SharedNetwork",
      extensible_attributes => { 'Site' => 'Santa Clara' }
    );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of DHCP a shared network object

authority( )

Use this method to set or retrieve the authority flag of a shared network object. This setting overrides the member-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the authority flag or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute is inherited from the member-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get authority
my $authority = $sharednetwork->authority()
# Modify authority
$sharednetwork->authority("true");
```

bootfile( )

Use this method to set or retrieve a bootfile attribute of a shared network object. You can configure the DHCP server to support clients that use the boot file name option in their DHCPREQUEST messages. This setting overrides the member-level settings. The default value is undefined, which indicates that this attribute is inherited from the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::SharedNetwork->bootserver() and Infoblox::DHCP::SharedNetwork->nextserver().

Parameter

The name of the file that the client must download. The file name must be in string format and can have a maximum of 128 characters. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get bootfile
my $bootfile = $sharednetwork->bootfile();
# Modify bootfile
$sharednetwork->bootfile("bootfile1");
#Un-override bootfile
$sharednetwork->bootfile(undef);
```

bootserver( )

Use this method to set or retrieve the bootserver attribute of a shared network object. You can specify the name and/or IP address of the boot server that the host must access in order to boot. This setting overrides the member-level settings. The default value is undefined, which indicates that this attribute is inherited from the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::SharedNetwork->bootfile() and Infoblox::DHCP::SharedNetwork->nextserver().

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get bootserver
my $bootserver = $sharednetwork->bootserver();
# Modify bootserver
$sharednetwork->bootserver("abc.domain.com");
#Un-override bootserver
$sharednetwork->bootserver(undef);
```

comment( )

Use this method to set or retrieve a descriptive comment for a shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $sharednetwork->comment();
# Modify comment
$sharednetwork->comment("add a shared network");
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname flag of a shared network object. Enable this method to allow only the DHCP server to generate host name. This setting overrides the member-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to generate the host name or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute is inherited from the member-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_generate_hostname
my $ddns_generate_hostname = $sharednetwork->ddns_generate_hostname()
# Modify ddns_generate_hostname
$sharednetwork->ddns_generate_hostname("true");
#Un-override
$sharednetwork->ddns_generate_hostname(undef);
```

ddns_server_always_updates( )

Use this method to set or retrieve the ddns_server_always_updates flag of a shared network object. Enable this method to allow only the DHCP server to update DNS,regardless of the requests from the DHCP clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to allow only the DHCP server to update DNS or "false" to deactivate/unset it. The default value for this field is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_server_always_updates
my $ddns_server_always_updates = $sharednetwork->ddns_server_always_updates()
# Modify ddns_server_always_updates
$sharednetwork->ddns_server_always_updates("true");
```

ddns_update_fixed_addresses( )

Use this method to set or retrieve the ddns_update_fixed_addresses flag of a shared network object. By default, the DHCP server does

not update DNS when it allocates a fixed address to a client. You can configure the DHCP server to update the A and PTR record of clients with fixed addresses. When you enable this feature and the DHCP server adds A and PTR records for a fixed address, the DHCP server never discards the records. This setting overrides the member-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to update DNS when the device allocates a fixed address to a client or "false" to deactivate/unset it. The default value for this field is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_update_fixed_addresses
my $ddns_update_fixed_addresses= $sharednetwork->ddns_update_fixed_addresses()
# Modify ddns_update_fixed_addresses
$sharednetwork->ddns_update_fixed_addresses("true");
#Un-override
$sharednetwork->ddns_update_fixed_addresses(undef);
```

ddns_ttl( )

Use this method to set or retrieve the DNS update Time to Live (TTL) value of a shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_ttl method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_ttl attribute to "false".

Note that when ddns_ttl is set to a defined value and override_dddns_ttl is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_ttl(12); $object->override_ddns_ttl("false"); will set override_ddns_ttl to "false", and the sequence $object->override_ddns_ttl("false"); $object->ddns_ttl(12); will result in override_ddns_ttl="true".

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration,in seconds, that the update is cached. Zero indicates that the update should not be cached. The default value is zero.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get DNS Update ttl
my $dns_ttl = $sharednetwork->ddns_ttl();
#Modify DNS Update ttl
$sharednetwork->ddns_ttl(1200);
```

ddns_use_option81( )

Use this method to set or retrieve the ddns_use_option81 flag of a shared network object. Set this flag to define the settings for option 81 at the network level. This setting overrides the member-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to configure support for option 81 or "false" to deactivate/unset it. The default value for this field is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ddns_use_option81
my $ddns_use_option81 = $sharednetwork->ddns_use_option81()
```

```
# Modify ddns_use_option81
$sharednetwork->ddns_use_option81("true");
# un-override ddns_use_option81
$sharednetwork->ddns_use_option81(undef);
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp flag of a DHCP Shared Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the deny_bootp flag or "false" to deactivate/unset it. The default value is undefined, which indicates that this attribute is inherited from the member-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $sharednetwork->deny_bootp();
#Modify deny_bootp
$sharednetwork->deny_bootp("true");
```

disable( )

Use this method to set or retrieve the disable flag of a shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag and disable the shared network or "false" to unset the flag and enable the shared network. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get disable
my $disable = $sharednetwork->disable()
# Modify disable
$sharednetwork->disable("true");
```

enable_ddns( )

Use this method to set or retrieve the dynamic DNS updates flag of a DHCP shared network object. The DHCP server can send DDNS updates to DNS servers in the same grid and to external DNS servers. This setting overrides the member-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the dynamic DNS updates flag or "false" to deactivate/unset it. The default value for this field is undefined, which indicates that this attribute is inherited from the member-level setting.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_ddns
my $enable_ddns = $sharednetwork->enable_ddns()
# Modify enable_ddns
$sharednetwork->enable_ddns("true");
# Un-override enable_ddns
$sharednetwork->enable_ddns(undef);
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a Shared Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $shared_network->extensible_attributes();
#Modify extensible attributes
$shared_network->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter',
'Tom' ] });
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the ignore_dhcp_option_list_request flag of a DHCP shared network object. If this flag is set to true all the defined DHCP options will be returned to the client. This setting overrides the member level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate/unset it. The default value is false which indicates that this attribute inherit the member level setting..

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $sharednetwork->ignore_dhcp_option_list_request();
# Un-override ignore_dhcp_option_list_request
$sharednetwork->ignore_dhcp_option_list_request(undef);
# Modify ignore_dhcp_option_list_request
$sharednetwork->ignore_dhcp_option_list_request("true");
```

lease_scavenge_time( )

Use this method to set or retrieve the lease_scavenge_time value of a Shared Network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_lease_scavenge_time method to "true". Setting the parameter to undefined causes the appliance to use the default from the upper level and automatically resets the override_lease_scavenge_time attribute to "false".

Note that when lease_scavenge_time is set to a defined value and override_lease_scavenge_time is set to "false", the last operation takes precedence. Thus the sequence $object->lease_scavenge_time(604800); $object->override_lease_scavenge_time("false"); will set override_lease_scavenge_time to "false", and the sequence $object->override_lease_scavenge_time("false"); $object->lease_scavenge_time(604800); will result in override_lease_scavenge_time="true".

Parameter

An integer that specifies the period of time, in seconds, that free and backup leases remain in the database before they are automatically deleted. To disable lease scavenging, set the parameter to -1. The minimum positive value must be greater than 86400 seconds (1 day). The default value of this parameter is -1, which means that lease scavenging is disabled.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lease_scavenge_time
my $lease_scavenge = $sharednetwork->lease_scavenge_time();
#Modify lease_scavenge_time
$sharednetwork->lease_scavenge_time(30 * 24 * 60 * 60);
```

name( )

Use this method to retrieve the name of a shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the shared network in string format. A name can have a maximum of 32 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $sharednetwork = $sharednetwork->name();
```

networks( )

Use this method to retrieve the DHCP networks attribute of a shared network object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Network objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $network1 = Infoblox::DHCP::Network->new(
                                        "network" => "20.0.0.0/8",
                                    );
my $network2 = Infoblox::DHCP::Network->new(
                                        "network" => "30.0.0.0/8",
                                    );

#Get networks
my $sharednetwork = $sharednetwork->networks();
#Modify networks
my $sharednetwork = $sharednetwork->networks([$network1, $network2]);
```

network_view( )

Use this method to set or retrieve the network view of the DHCP shared network.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the DHCP shared network is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $sharednetwork->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$sharednetwork->network_view($nview);
```

nextserver( )

Use this method to set or retrieve the nextserver attribute of a shared network object. You can specify the name and/or IP address of the next server that the host needs to access in order to boot. This setting overrides the member-level settings. The default value is undefined which indicates that this attribute is inherited from the member-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::DHCP::SharedNetwork->bootfile() and Infoblox::DHCP::SharedNetwork->bootserver().

Parameter

The next server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A next server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get nextserver
my $nextserver = $sharednetwork->nextserver();
# Modify nextserver
$sharednetwork->nextserver("blue.domain.com");
#Un-override nextserver
$sharednetwork->nextserver(undef);
```

options( )

Use this method to set or retrieve the options value of a member DHCP server. DHCP options provide network configuration settings and various services available on the network for DHCP. This setting overrides the member-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $sharednetwork->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name  => "lease-time",
    value => 50
);

$sharednetwork->options([$option1]);
```

override_ddns_ttl( )

The override_ddns_ttl attribute controls whether the ddns_ttl value in the DHCP shared network object is used, instead of the grid default.

The override_ddns_ttl attribute can be specified explicitly. It is also set implicitly when ddns_ttl is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_ttl. Set the parameter to "false" to inherit the grid-level setting for ddns_ttl.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_ttl
my $override_ddns_ttl=$sharednetwork->override_ddns_ttl( );
#Modifying override_ddns_ttl
$sharednetwork->override_ddns_ttl("true");
```

override_lease_scavenge_time( )

The override_lease_scavenge_time attribute controls whether the lease_scavenge_time value in the shared network is used, instead of the upper-level default.

The override_lease_scavenge_time attribute can be specified explicitly. It is also set implicitly when lease_scavenge_time is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for lease_scavenge_time. Set the parameter to "false" to inherit the setting for lease_scavenge_time from the upper level.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_lease_scavenge_time
my $override_lease_scavenge_time=$sharednetwork->override_lease_scavenge_time( );
#Modifying override_lease_scavenge_time
$sharednetwork->override_lease_scavenge_time("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the DHCP shared network object is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$sharednetwork->override_update_dns_on_lease_renewal( );
```

```
#Modifying override_update_dns_on_lease_renewal
$sharednetwork->override_update_dns_on_lease_renewal("true");
```

pxe_lease_time( )

Use this method to set or retrieve the PXE lease time value of a shared network object. Some hosts use PXE (Preboot Execution Environment) to boot remotely from a server. To better manage your IP resources, set a different lease time for PXE boot requests. You can configure the DHCP server to allocate an IP address with a shorter lease time to hosts that send PXE boot requests, so IP addresses are not leased longer than necessary. This setting overrides the member-level settings. The default value is undefined, which indicates that this attribute is inherited from the member-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the update is cached. Zero indicates that the update is not cached. The default value is undefined

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get PXE Lease Time
my $pxe_lease_time = $sharednetwork->pxe_lease_time();
#Modify PXE Lease Time
$sharednetwork->pxe_lease_time(1100);
#Un-override PXE Lease Time
$sharednetwork->pxe_lease_time(undef);
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$sharednetwork->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$sharednetwork->update_dns_on_lease_renewal("true");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP shared network object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
```

```
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP network object

```
my $network1 = Infoblox::DHCP::Network->new(
                                    "network" => "10.0.0.0/8",
                                );
my $network2 = Infoblox::DHCP::Network->new(
                                    "network" => "11.0.0.0/8",
                                );

my $response = $session->add($network1)
or die("Add \"10.0.0.0/8\"Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network \"10.0.0.0/8\"added successfully\n";

my $response = $session->add($network2)
       or die("Add  \"11.0.0.0\"Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network  \"11.0.0.0/8\",added successfully\n";
```

#Create a DHCP shared network object

```
my $sharednetwork1 = Infoblox::DHCP::SharedNetwork->new(
                                    "name"                      => "myname",
                                    "comment"                   => "add shared network",
                                    "networks"                  => [ $network1 , $network2 ],
                                    "disable"                   => "false",
                                    "authority"                 => "true",
                                    "enable_ddns"               => "true",
                                    "ddns_use_option81"         => "true",
                                    "ddns_generate_hostname"    => "true",
                                    "ddns_server_always_updates" => "true",
                                    "ddns_update_fixed_addresses" => "true",
                                    "pxe_lease_time"            => "3600",
                                    "bootserver"                => "someserver.com",
                                    "nextserver"                => "3.3.3.3",
                                    "bootfile"                  => "bootfile.com",


                                    );

my $response = $session->add( $sharednetwork1 )
or die("Add Shared Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Shared Network added successfully\n";
```

#Search for a specific shared network object

```
#Search the network
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::SharedNetwork",
    "name"  => "my.*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search Shared Network failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search Shared Network found at least 1 matching entry\n";
```

#Get and modify a shared network attribute

```
#GetShared Network through the session
my  @retrieved_objs = $session->get(
```

```
            object =>"Infoblox::DHCP::SharedNetwork" ,
            "name" => "myname"
        );

 my $object = $retrieved_objs[0];

 unless ($object) {
     die("Get Shared Network object failed: ",
         $session->status_code() . ":" . $session->status_detail());
 }
 print "Get Shared Network found at least 1 matching entry\n";

 #Modify attributes of the specified Network
 $object->disable("true"); #Apply the changes
 my $res = $session->modify($object) ;
 if ( !$res)
 {
     print "Modify Shared Network attribute failed: ".$session->status_code() . ":" . $session-
>status_detail();
 }
 else {
 print "Shared Network object modified successfully \n";
 }

 #Un-override bootfile
 $object->bootfile(undef);

 #Un-override bootserver
 $object->bootserver(undef);

 #Un-override nextserver
 $object->nextserver(undef);
```

#Remove a shared network object

```
  #Get Network object through the session
  my @retrieved_objs = $session->get(
            object  => "Infoblox::DHCP::SharedNetwork" ,
            name    => "myname"
     );

    my $object = $retrieved_objs[0];

    unless ($object) {
     die("Get Shared Network object failed: ",
         $session->status_code() . ":" . $session->status_detail());
     }

 print "Get Shared Network object found at least 1 matching entry\n";

 #Submit the shared network object for removal

 $session->remove($object)
     or die("Remove Shared Network failed: ",
         $session->status_code() . ":" . $session->status_detail());

 print "Shared Network removed successfully \n";

 #Submit the network object for removal
 my $response = $session->remove($network1)
 or die("Remove \"10.0.0.0/8\"Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

 print "Network \"10.0.0.0/8\"remove successfully\n";

 #Submit the network object for removal
 my $response = $session->remove($network2)
        or die("Remove  \"11.0.0.0\"Network failed: ",
             $session->status_code() . ":" . $session->status_detail());

 print "Network \"11.0.0.0/8\"remove successfully\

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::Network, Infoblox::DHCP::View, Infoblox::DHCP::Option, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::SplitNetwork - Split network object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
- METHODS
  - add_all_subnetworks( )
  - auto_create_reversezone( )
  - network( )
  - prefix( )
  - prefix_collector_ipv6_network_addr( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::SplitNetwork - Split network object.

DESCRIPTION

Once a network is created for DHCP, smaller subnetworks can be configured with larger netmasks. A larger netmask defines a larger number of network addresses and a smaller number of host addresses. Using Infoblox split network feature, the parent network can be split into multiple smaller networks without configuring each of the newly-created networks.

CONSTRUCTOR

```
my $split_network = Infoblox::DHCP::SplitNetwork->new(
            network                 => $Network|$IPv6Network,   #Required
            prefix                  => $num,                    #Required
            add_all_subnetworks     => "true" | "false",        #Optional / Default is "false"
            auto_create_reversezone => "true" | "false",        #Optional / Default is "false"
            prefix_collector_ipv6_network_addr => $string,      #Optional / Default is undefined
);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that can be applied to a split network object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $network = Infoblox::DHCP::Network->new(
                            network => "10.0.0.0/8",
                            comment => "add network",
                            );

my $split_network = Infoblox::DHCP::SplitNetwork->new (
                      network                 => $network,
                      prefix                  => 12,
                      add_all_subnetworks     => "true",
                      );
#Submit for addition
my $response = $session->add( $split_network );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute value of a split network object.

add_all_subnetworks( )

Use this method to set the add_all_subnetworks flag. If this flag is set, then all possible subnetworks will be added otherwise only networks with fixed addresses will be added.

Include the specific parameter to set the attribute value.

Parameter

Specify "true" to set add_all_subnetworks flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Set add_all_subnetworks
$split_network->add_all_subnetworks("true");
```

auto_create_reversezone( )

Use this method to set the auto_create_reversezone flag. When set, this flag will automatically create reverse-mapping zones for the subnets.

Include the specified parameter to set the attribute value.

Parameter

Specify "true" to set auto_create_reversezone flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Set auto_create_reversezone
$split_network->auto_create_reversezone("true");
```

network( )

Use this method to specify the network to be split.

Include the specified parameter to set the attribute value.

Parameter

The valid values are Infoblox::DHCP::Network or Infoblox::DHCP::IPv6Network objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Specify network
my $network = Infoblox::DHCP::Network->new(
                    network => "10.0.0.0/8",
                    comment => "add network",
                    );

$split_network->network($network);

#Specify ipv6 network
my $network = Infoblox::DHCP::IPv6Network->new(
                    network => "10::/48",
                    comment => "add network",
                    );

$split_network->network($network);
```

prefix( )

Use this method to specify the appropriate subnet mask for each subnet created after splitting the network.

Include the specified parameter to set the attribute value.

Parameter

Number with the subnet mask value. This value must be 1 greater than the subnet mask of the network to be split. The maximum value is 31 for IPv4 network or 127 for IPv6 network.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Specify prefix
$split_network->prefix(16);
```

prefix_collector_ipv6_network_addr( )

Use this method to specify the network that contains the DHCP IPv6FixedAddress and IPv6Range objects whose address_type is 'PREFIX' after a network is split.

Include the specified parameter to set the attribute value.

Parameter

The valid value is an IPv6 network address in string format and without any prefix bits appended.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Set prefix_collector_ipv6_network_addr( )
$split_network->prefix_collector_ipv6_network_addr("2001::");
```

SAMPLE CODE

The following sample code demonstrates the split network operation. Also this sample code includes error handling for the operations.

#Preparation prior to splitting a network

```
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully.\n";

#Create network prior to splitting the network
my $network = Infoblox::DHCP::Network->new(
                            network => "20.0.0.0/8",
                            comment => "add network",
                            );
unless($network){
       die("Construct network object fail: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
       }

print"Network object created successfully.\n";

#Verify if the network exists
my $object = $session->get(
               object  => "Infoblox::DHCP::Network",
               network => "20.0.0.0/8"
               );
unless($object){
       print"Network does not exist on server, safe to add the network.\n";
       $session->add($network)
               or die("Add network failed: ",
                       $session->status_code(). ":" .$session->status_detail());
       }
```

```
print"Network added successfully.\n";
```

#Split the network

```
my $split_network = Infoblox::DHCP::SplitNetwork->new (
                        network                 => $network,
                        prefix                  => 16,
                        add_all_subnetworks     => "true",
                        auto_create_reversezone => "true",
                      );
unless($split_network){
       die("Construct split network object fail: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
       }

print"Split network object created successfully.\n";

#Split the network
$session->add( $split_network )
         or die("Split network fail:",
                    $session->status_code(). ":" .$session->status_detail());

print"Splited network successfully.\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::DHCP::Network

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Statistics - Manages DHCP statistics.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
- METHODS
  - available_hosts( )
  - dynamic_hosts( )
  - dhcp_utilization_status( )
  - static_hosts( )
  - usage( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Statistics - Manages DHCP statistics.

DESCRIPTION

The DHCP statistics object is used to view the DHCP statistics of the network in an Infoblox appliance.

CONSTRUCTOR

The Infoblox::DHCP::Statistics object is automatically generated upon the successful creation of the Infoblox::DHCP::Network object. It does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to a DHCP statistics object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get the DHCP statistics:

statistics_object - Required. An object returned via Infoblox::Session->get() or Infoblox::Session-
>search(). Supported objects are Infoblox::DHCP::Network, Infoblox::DHCP::Range,
Infoblox::DHCP::SharedNetwork, and Infoblox::DHCP::MSSuperscope.
```

Example

```
# API can return more than 1 answer for DHCP::Statistics
# for each member of the network
my @network_stats = $session->get(
        object            => "Infoblox::DHCP::Statistics",
        statistics_object => $network);
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of a DHCP statistics object.

available_hosts( )

Use this method to retrieve the number of available IP addresses in the given network.

Parameter

none

Returns

The method returns the attribute value.

Example

```
#Get available_hosts
my $available_hosts = $network_stats->available_hosts();
```

dynamic_hosts( )

Use this method to retrieve number of hosts which are assigned dynamic IP address.

Parameter

none

Returns

The method returns the attribute value.

Example

```
#Get dynamic_hosts
my $dynamic_hosts = $network_stats->dynamic_hosts();
```

dhcp_utilization_status( )

Use this method to retrieve the DHCP utilization status.

Parameter

none

Returns

The method returns the attribute value. The returned value is one of "FULL", "HIGH", "LOW" or "NORMAL".

Example

```
#Get member
my $status = $network_stats->dhcp_utilization_status();
```

static_hosts( )

Use this method to retrieve the number of hosts with fixed addresses.

Parameter

none

Returns

The method returns the attribute value.

Example

```
#Get static_hosts
my $static_hosts = $network_stats->static_hosts();
```

usage( )

Parameter

none

Returns

The method returns the attribute value.

Example

```
#Get usage
my $usage = $network_stats->usage();
```

SAMPLE CODE

The following sample code demonstrates the get session method on a DHCP statistics object. Also, this sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;
```

#Preparation prior to getting DHCP statistics

```
#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $member = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
);

unless($member) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create a Network View object
my $network_view = Infoblox::DHCP::View->new(
                   name => "my_network_view",
);
unless($network_view) {
            die("Construct Network View object failed: ",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network View object created successfully\n";

#Add the Network View object into the Infoblox appliance through a session
$session->add($network_view)
     or die("Add Network View object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Network View added successfully\n";

#Create the Network object with this member
my $network = Infoblox::DHCP::Network->new(
                   network => "1.0.0.0/255.0.0.0",
                   members => [ $member ],
                   network_view => $network_view,
);
unless($network) {
            die("Construct Network object failed: ",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
     or die("Add Network object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Network added successfully\n";

#Create the DHCP Range object with this member
my $dhcp_range = Infoblox::DHCP::Range->new(
    network    => "1.0.0.0/8",
    start_addr => "1.0.0.1",
    end_addr   => "1.0.0.10",
    member     => $member,
    network_view => $network_view,
);
unless($dhcp_range) {
     die("Construct DHCP Range failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP range object created successfully\n";
```

```
$session->add($dhcp_range)
     or die("Add DHCP range object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DHCP range added successfully\n";

#Create the Fixed Address object
my $fixed_address = Infoblox::DHCP::FixedAddr->new(
                                                "network" => "1.0.0.0/8",
                                                "ipv4addr" => "1.0.0.2",
                                                "mac" => "22:22:22:22:22:22",
                                                "network_view" => $network_view,
                                                );
unless($fixed_address) {
     die("Construct DHCP Fixed address failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Fixed address object created successfully\n";

$session->add($fixed_address)
     or die("Add DHCP Fixed address object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DHCP Fixed address added successfully\n";

#Get the DHCP statistics

 my  @result_array = $session->get(
     object  => "Infoblox::DHCP::Network",
     network => "1.0.0.0/8",
     network_view => "my_network_view");

 my $network = $result_array[0];
 unless ($network) {
     die("Get Network failed: ",
         $session->status_code() . ":" . $session->status_detail());
 }

 @result_array = $session->get(
                              object  => "Infoblox::DHCP::Statistics",
                              statistics_object  => $network,
                             );

 my $network_stats = $result_array[0];
 unless ($network_stats) {
     die("Get Network statistics failed: ",
         $session->status_code() . ":" . $session->status_detail());
 }

 print "Get Network Statistics successful\n";

#Clean up # Remove the Network Statistics object that was just retrieved $session->remove($network_stats);

 # Retrieve and remove the Network View object
 my $network_view = $session->get(
         object => "Infoblox::DHCP::View",
         name => "my_network_view",
 );
 unless ($network_view) {
      die("Get Network View failed: ",
            $session->status_code() . ":" . $session->status_detail());
 }
 $session->remove($network_view);

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::Network, Infoblox::DHCP::View, Infoblox::DHCP::FixedAddr, Infoblox::DHCP::Range, Infoblox::Session->add(), Infoblox::Session->get()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::Template - DHCP Template object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::NetworkTemplate->range_templates( )
  - Infoblox::DHCP::NetworkTemplate->fixed_address_templates( )
- METHODS
  - name( )
  - count( )
  - offset( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::Template - DHCP Template object.

DESCRIPTION

The DHCP associates Infoblox::DHCP::RangeTemplate->offset(), Infoblox::DHCP::RangeTemplate->number_of_addresses() and Infoblox::DHCP::FixedAddrTemplate->offset(), Infoblox::DHCP::FixedAddrTemplate->number_of_addresses() with Infoblox::DHCP::NetworkTemplate. When optional fields offset/count are set in this object which is passed to Infoblox::DHCP::NetworkTemplate->fixed_address_templates() or Infoblox::DHCP::NetworkTemplate->range_templates() they are used instead of offset/number_of_addresses values in Infoblox::DHCP::RangeTemplate and Infoblox::DHCP::FixedAddrTemplate when creating Infoblox::DHCP::Network based on a template.

CONSTRUCTOR

```
my $template = Infoblox::DHCP::Template->new(
      name                 => $string,       #Required
      count                => $num,          #Optional
      offset               => $num,          #Optional
);
```

MODULE METHODS

The following functions are available to be applied to a DHCP template object.

Infoblox::DHCP::NetworkTemplate->range_templates( )

Use this function to specify range templates for the DHCP network template object. See Infoblox::DHCP::NetworkTemplate->range_templates() for parameters and return values.

Example

```
          #Construct and add a DHCP Range Template object
          my $range_template = Infoblox::DHCP::RangeTemplate->new(
                   name                 => "custom_range_template",
                   offset               => "20", # this won't be used if specified in
     Infoblox::DHCP::Template
                   number_of_addresses  => "10", # this won't be used if specified in
     Infoblox::DHCP::Template
          );
          $session->add($range_template);

          #Construct a DHCP Template object
          my $template = Infoblox::DHCP::Template->new(
                   offset               => "10", # this will be used instead if specified
                   count                => "5", # this will be used instead if specified
                   name                 => "custom_range_template"
          );

          # Configure DHCP Template on the DHCP Network template object
          $network_template->range_templates([$template]);
```

Infoblox::DHCP::NetworkTemplate->fixed_address_templates( )

Use this function to specify fixed address templates for the DHCP network template object. See Infoblox::DHCP::NetworkTemplate->fixed_address_templates() for parameters and return values.

Example

```
#Construct and add a DHCP Fixed Address Template object
my $fixed_address_template = Infoblox::DHCP::FixedAddrTemplate->new(
        name                => "custom_fixed_address_template",
        offset              => "20", # this won't be used if specified in
Infoblox::DHCP::Template
        number_of_addresses  => "10", # this won't be used if specified in
Infoblox::DHCP::Template
);
$session->add($fixed_address_template);

#Construct a DHCP Template object
my $template = Infoblox::DHCP::Template->new(
        offset              => "10", # this will be used instead if specified
        count               => "5", # this will be used instead if specified
        name                => "custom_fixed_address_template"
);

# Configure DHCP Template on the DHCP Network template object
$network_template->fixed_address_templates([$template]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP range template object.

name( )

Use this method to set or retrieve name of Infoblox::DHCP::RangeTemplate or Infoblox::DHCP::FixedAddrTemplate object assigned to this template object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of range template or fixed address template object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $template->name();
#Modify name
$template->name("custom_range_or_fixed_address_template_name");
```

count( )

Use this method to set or retrieve the number of addresses of the DHCP range or fixed address template. This value will be used instead of Infoblox::DHCP::RangeTemplate->number_of_addresses() or Infoblox::DHCP::FixedAddrTemplate->number_of_addresses() if specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of addresses in this template.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get count
my $count = $template->count();
```

```
#Modify count
$template->count("10");
```

offset( )

Use this method to set or retrieve the address offset of the DHCP range or fixed address template. This value will be used instead of Infoblox::DHCP::RangeTemplate->offset() or Infoblox::DHCP::FixedAddrTemplate->offset() if specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The address offset of this template.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get offset
my $offset = $template->offset();
#Modify offset
$template->offset("10");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DHCP template object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a DHCP Range object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;
#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
   master   => "192.168.1.2",
   username => "admin",
   password => "infoblox"
);

unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create the DHCP Range template object

```
my $dhcp_range_template = Infoblox::DHCP::RangeTemplate->new(
        name                   => "custom_range_template",
        offset                 => "10",
        number_of_addresses     => "10",
);
unless($dhcp_range_template) {
       die("Construct DHCP Range template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Range template object created successfully\n";

#Add the DHCP Range template object into the Infoblox appliance through a session
$session->add($dhcp_range_template)
    or die("Add DHCP Range template object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Range template object added to Infoblox appliance successfully\n";
```

#Create the Fixed Address template object

```
my $dhcp_fixed_address_template = Infoblox::DHCP::FixedAddrTemplate->new(
        name                   => "custom_fixed_address_template",
        offset                 => "10",
        number_of_addresses     => "10",
);
```

```
unless($dhcp_fixed_address_template) {
       die("Construct DHCP Fixed Address template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Fixed Address template object created successfully\n";

#Add the DHCP Fixed Address Template  object into the Infoblox appliance through a session
$session->add($dhcp_fixed_address_template)
    or die("Add DHCP Fixed Address template object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Fixed Address template object added to Infoblox appliance successfully\n";
```

#Create DHCP template objects

```
#Construct a DHCP Template object
my $tmpl1 = Infoblox::DHCP::Template->new(
         offset               => "10", # this will be used instead if specified
         count                => "5", # this will be used instead if specified
         name                 => "custom_range_template"
);

unless($tmpl1) {
       die("Construct DHCP template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP template object created successfully\n";

#Construct a DHCP Template objects
my $tmpl2 = Infoblox::DHCP::Template->new(
         offset               => "20", # this will be used instead if specified
         count                => "3", # this will be used instead if specified
         name                 => "custom_fixed_address_template"
);

unless($tmpl2) {
       die("Construct DHCP template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP template object created successfully\n";
```

#Add DHCP template to DHCP Network Template and insert it

```
my $network_template = Infoblox::DHCP::NetworkTemplate->new(
        name                   => "network_template",
        netmask                => "255.0.0.0",
        range_templates         => [ $tmpl1 ],
        fixed_address_templates => [ $tmpl2 ],
);
unless($network_template) {
       die("Construct DHCP Network template object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Network template object created successfully\n";

#Add the DHCP Network template object into the Infoblox appliance through a session
$session->add($network_template)
    or die("Add Network template object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DHCP Network template object added to Infoblox appliance successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::RangeTemplate, Infoblox::DHCP::FixedAddrTemplate, Infoblox::DHCP::NetworkTemplate, Infoblox::DHCP::Network

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DHCP::View - DHCP Network View object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - comment( )
  - extensible_attributes( )
  - external_ddns_primaries( )
  - internal_ddns_zones( )
  - internal_updates_view( )
  - associated_dns_views( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DHCP::View - DHCP Network View object

DESCRIPTION

Network view provides the ability to manage multiple views of IP address spaces from NIOS. These address spaces could be overlapping or non-overlapping. When the network view is not specified during a write operation, the objects are automatically associated with the system-defined default network view.

A network view is a single routing domain, with its own networks and shared networks. You can manage the networks in one network view independently of the other network views.

CONSTRUCTOR

```
  my $network = Infoblox::DHCP::View->new(
                 name                          => $string,                        #Required
                 comment                       => $string,                        #Optional / Default is
empty
                 extensible_attributes          => { $string => $string|$num, $string => [ $string|$num,
...], ... },                    #Optional / Default is undefined
                 external_ddns_primaries        => [ $Nameserver1,$Nameserver2,...],#Optional / List of
remote zone assignments
                 internal_ddns_primaries        => [ $ForwardZone1,$ForwardZone2,...],#Optional / List of
forward mapping zones in grid
               );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that can apply to a DHCP Network View object.

Infoblox::Session->add( )

Use this method to add a network view object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $network_view = Infoblox::DHCP::View->new(
       name => "netview",
);

# Submit for adding network view
my $response = $session->add( $network_view );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching network view objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DHCP Network View object:

 name                  - Required. A name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
 my  @result_array = $session->get(
      object        => "Infoblox::DHCP::View" ,
      name          => "netview"
 );

# get all DHCP view objects with a given extensible attribute
my @retrieved_objs = $session->get(
   object => "Infoblox::DHCP::View",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a network object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify comment method.
$object->comment("My modified comment");
# Submit modification
my $response = $session->modify( $object );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my  @result_array = $session->get(
      object        => "Infoblox::DHCP::View" ,
      name          => "netview"
 );

# find the desired object from the retrieved list.
my $desired_network_view = $result_array[0];

# Submit for removal
my $response = $session->remove( $desired_network_view );
```

Infoblox::Session->search( )

Use this method to search for a DHCP Network View object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DHCP Network View object:

 name                  - Required. A name in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for DHCP Network View object with name 'netview'
    my @retrieved_objs = $session->search(
      object       => "Infoblox::DHCP::Network",
      name         => "^netview\$"
    );

# search for DHCP views with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
```

```
    object => "Infoblox::DHCP::View",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a DHCP Network View object

name( )

Use this method to set or retrieve the name of a DHCP Network View object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the network view is in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $name = $network_view->name()
# Modify name
$network_view->name("netview_renamed");
```

comment( )

Use this method to set or retrieve a descriptive comment of a DHCP Network View object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the network view in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $network_view->comment();
# Modify comment
$network_view->comment("add a network view");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DHCP view object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $firstview->extensible_attributes();
```

```
#Modify extensible attributes
$firstview->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

external_ddns_primaries( )

Use this method to set or retrieve the network view-level external_ddns_primaries value. This specifies the external zone that is updated dynamically. The DHCP server can send dynamic updates to a specified external name server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A valid value is an array reference that contains Infoblox::DNS::Nameserver object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get external_ddns_primaries
my $external_ddns_primaries = $network_view->external_ddns_primaries();

# Modify external_ddns_primaries
my $nameserver1 = Infoblox::DNS::Nameserver->new(
       ddns_zone => "ns1.del.com",
       ipv4addr  => "11.11.11.11",
   );
my $nameserver2 = Infoblox::DNS::Nameserver->new(
        ddns_zone => "3.2.1.in-addr.arpa",
        ipv4addr  => "22.22.22.22",
   );
$network_view->external_ddns_primaries([$nameserver1, $nameserver2]);

# Clear external_ddns_primaries list
$network_view->external_ddns_primaries([]);
```

internal_ddns_zones( )

Use this method to set or retrieve forward mapping zones which are being served in the grid. Only an authoritative zone can be assigned to this list. All other types of zones will generate an error.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A valid value is an array reference that contains Infoblox::DNS::Zone object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get internal_ddns_zones
my $internal_ddns_zones = $network_view->internal_ddns_zones();

# Modify internal_ddns_zones
my $nameserver1 = Infoblox::DNS::Nameserver->new(
       name     => "ns1.del.com",
       ipv4addr => "11.11.11.11",
   );
my $nameserver2 = Infoblox::DNS::Nameserver->new(
       name     => "test",
       ipv6addr => "2006::b671",
   );
my $zone1 = Infoblox::DNS::Zone->new(
       name     => "test1.com",
       primary  => [$nameserver1],
       secondaries => [$nameserver2],
   );
my $zone2 = Infoblox::DNS::Zone->new(
       name     => "test2.com",
```

```
    );
$network_view->internal_ddns_zones([$zone1, $zone2]);

# Clear internal_ddns_zones list
$network_view->internal_ddns_zones([]);
```

internal_updates_view( )

Use this method to specify or retrieve the internal updates DNS view value of a network view. When you enable the appliance to send updates to grid members, you must specify which DNS view receives the updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A valid value is an Infoblox::DNS::View object for internal dynamic updates.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get internal_updates_view
my $internal_updates_view = $network_view->internal_updates_view();
# Modify internal_updates_view
my $otherview = Infoblox::DNS::View->new(
       name => "other",
    );
$network_view->internal_updates_view($otherview);
```

associated_dns_views( )

Use this method to retrieve the list of DNS views associated with this network view. This list is read-only and cannot be set.

Parameter

None

Returns

Returns an array reference that contains Infoblox::DNS:View object(s).

Example

```
# Get associated_dns_views
my $associated_dns_views = $network_view->associated_dns_views()
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DHCP Network View insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DHCP Network View object

```
my $network_view = Infoblox::DHCP::View->new(
```

```
            name     => "netview",
           );

my $response = $session->add($network_view)
       or die("Add Network View failed: ",
             $session->status_code() . ":" . $session->status_detail());

print "Network View added successfully\n";

B<#Search for a specific Network View>

#Search the network view
my @retrieved_objs = $session->search(
    object  => "Infoblox::DHCP::View",
    name => '^netview$',
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search Network View failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search Network found at least 1 matching entry\n";
```

#Get and modify a Network View attribute

```
#Get Network View through the session
my  @retrieved_objs = $session->get(
           object => "Infoblox::DHCP::View" ,
           name   => "netview",
        );

my $object = $retrieved_objs[0];

unless ($object) {
    die("Get Network View object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network View found at least 1 matching entry\n";

#Modify one of the attributes of the specified Network View attribute
$object->comment("updated comment"); #Apply the changes

$session->modify($object)
    or die("Modify Network View attribute failed: ",
           $session->status_code() . ":" . $session->status_detail());

print "Network View object modified successfully \n";
```

#Remove a Network View Object

```
 #Get Network View object through the session
 my @retrieved_objs = $session->get(
           object  => "Infoblox::DHCP::View" ,
           name    => "netview",
       );

   my $object = $retrieved_objs[0];

   unless ($object) {
    die("Get Network View object failed: ",
        $session->status_code() . ":" . $session->status_detail());
    }

print "Get Network View object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove Network View failed: ",
        $session->status_code() . ":" . $session->status_detail());

print "Network View removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DNS::View, Infoblox::DHCP::Network, Infoblox::DNS::Zone, Infoblox::DNS::Nameserver, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::AllRecords - A synthetic object used to return all record object types belonging to a zone.

- NAME
- DESCRIPTION
- SESSION METHODS
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - name( )
  - record( )
  - ttl( )
  - type( )
  - view( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::AllRecords - A synthetic object used to return all record object types belonging to a zone.

DESCRIPTION

By searching on this object, all the record objects associated with a specific zone can be retrieved

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS AllRecords object.

Infoblox::Session->search( )

Use this method to search for record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS AllRecords object:

 zone       - Required. A zone name in FQDN format.
 view       - Optional. The DNS view in which the records are located. By default, the default DNS
views is searched.
```

Example

```
# search for all record objects that match "domain.com" in the default DNS view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::AllRecords",
    name   => 'domain.com',
);
```

METHODS

This section describes all the methods that you can use to retrieve the attribute values of an AllRecords object. All methods are read-only and cannot be set.

comment( )

Use this method to retrieve the descriptive comment of the record associated with the AllRecords object.

Returns

The method returns the attribute value.

Example

```
#Get comment
```

```
my $comment = $allrecord->comment();
```

disable( )

Use this method to retrieve the disable flag of of the record associated with the AllRecords object (if present).

Returns

The method returns the attribute value.

Example

```
#Get disable
my $disable = $allrecord->disable();
```

name( )

Use this method to retrieve the name of of the record associated with the AllRecords object. Note that this value could differ from the value of the associated record's name field.

Returns

The method returns the attribute value.

Example

```
#Get name
my $name = $allrecord->name();
```

record( )

Use this method to retrieve the record object associated with the AllRecords object.

This object is considered a 'partial' object. See Infoblox::Session->fill_partial_object() for more information on how it can be converted to a 'full' API object.

If the AllRecords object refers to a record not currently supported in the Perl API, the member value will be undefined.

Returns

The method returns the attribute value.

Example

```
#Get record
my $record = $allrecord->record();
```

ttl( )

Use this method to retrieve the TTL value of of the record associated with the AllRecords object (if present).

Returns

The method returns the attribute value.

Example

```
#Get ttl
my $ttl = $allrecord->ttl();
```

type( )

Use this method to retrieve the type of the record associated with the AllRecords object. This is a descriptive string that identifies the record to which this AllRecords object refers. (For example, 'A record', 'MX record', ...)

Returns

The method returns the attribute value.

Example

```
#Get type
my $type = $allrecord->type();
```

view( )

Use this method to retrieve the DNS view name of the record associated with the AllRecords object.

Returns

The method returns the attribute value.

Example

```
#Get view
my $view = $allrecord->view();
```

zone( )

Use this method to retrieve the zone name of the record associated with the AllRecords object.

Returns

The method returns the attribute value.

Example

```
#Get zone
my $zone = $allrecord->zone();
```

SAMPLE CODE

In cases where a large number of matching records exist on the appliance, only a limited number will be returned
at a time. If all of the matching records are required, please consider using the paging support via Infoblox::Cursor.

```
#Code sample fragment Starts:

   $cursor=$session->new_cursor(
                               fetch_size => 100,
                               object => "Infoblox::DNS::AllRecords",
                               zone   => 'domain.com',
                               view   => 'default',
                              );

   while ($current_record = $cursor->fetch()) {
       # $current_record will contain a matching AllRecords object
   }

   # All matching AllRecords will be returned
   # when the cycle ends.
```

If you want full record objects for some of the above, you can add the following code

```
#Code sample fragment Starts:

    if( ... some criteria ...) {
        push @temp, $current_record->record();
    }

####Code Fragment Ends####
```

inside the while() cycle and afterwards call

```
#Code sample fragment Starts:

@full_record_objects=$session->fill_partial_object(\@temp);

####Code Fragment Ends####
```

to fill the wanted objects.

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->search()/>

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::BulkHost - Create a group of hosts.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - end_addr( )
  - name_template( )
  - prefix( )
  - reverse( )
  - start_addr( )
  - ttl( )
  - use_name_template( )
  - views( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::BulkHost - Create a group of hosts.

DESCRIPTION

If you need to add a large number of hosts, you can have the Infoblox appliance add them as a group and automatically assign host names based on a range of IP addresses and name format applied to it. This group of hosts is referred to as a BulkHost. The Infoblox appliance uses the name space bulk-xx-xx-xx-xx for BulkHosts, so this name should not be used for CNAMEs and host aliases because doing so causes conflicts. Before adding a BulkHost, make sure that no CNAMEs or host aliases uses this name.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
my $bulkhost = Infoblox::DNS::BulkHost->new(
    end_addr           => $ipv4addr,              #Required
    prefix             => $string,                #Required
    start_addr         => $ipv4addr,              #Required
    zone               => $fqdn,                  #Required
    comment            => $string,                #Optional / Default is empty
    disable            => "true" | "false",       #Optional / Default "false"
    name_template      => $string | undef,        #Optional / Default is undefined
    reverse            => "true" | "false",       #Optional / Default "false"
    ttl                => $num | undef,           #Optional / Default is undefined
    use_name_template  => "true" | "false",       #Optional / Default is "false"
    views              => [$View1]                        #Optional / Default is "default" view
);
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS BulkHost record object.

Infoblox::Session->add( )

Use this method to add a BulkHost record object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $bulk_host = Infoblox::DNS::BulkHost->new(
    zone               => "domain.com",
    comment            => "add bulk host",
    prefix             => "prefix",
    start_addr         => "1.0.0.1",
    end_addr           => "1.0.0.5",
    reverse            => "true",
    use_name_template  => "true",
    name_template      => "Three Octets"
);
#Submit for addition
my $response = $session->add($bulk_host)
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS BulkHost object:

 zone - Optional. A zone name in string format.
 prefix     - Optional. Prefix of the bulk host.
 start_addr - Optional. Start address of the bulk host.
 end_addr   - Optional. End address of the bulk host.
 comment    - Optional. A comment in string format.
 view - Optional. The DNS view in which the bulk host record is located. The default is all DNS
views.
 ttl - Optional. Time to Live (TTL) value in integer format.
 name_template - Optional. The bulk host name template in string format.
 disable - Optional. Disable flag of a DNS BulkHost record in boolean format.
 use_name_template - Optional. Use name template flag of a DNS BulkHost record in boolean format.
 reverse - Optional. Reverse flag of a DNS BulkHost record in boolean format.
```

Example

```
#Get BulkHost object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::BulkHost",
    zone   => "domain.com",
    prefix => "prefix",
    start_addr => "1.0.0.2",
    end_addr => "1.0.0.6".
    view => "default",
    ttl => 3600,
    name_template => "Three Octets",
    disable => "false",
    use_name_template => "true",
    reverse => "true"
);
```

Infoblox::Session->modify( )

Use this method to modify a BulkHost record object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the prefix.
$bulk_host->prefix("test");
#Submit modification
my $response = $session->modify( $bulk_host );
```

Infoblox::Session->remove( )

Use this method to remove a BulkHost record object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific BulkHost record object, and then submit this object for removal.

Example

```
#Get BulkHost object through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DNS::BulkHost",
    zone       => "domain.com",
```

```
        prefix     => "test",
        start_addr => "1.0.0.1",
        end_addr   => "1.0.0.5"
    );
    #Find the desired object from the retrieved list.
    my $desired_bulk = $retrieved_objs[0];
    #Submit for removal
    my $response = $session->remove( $desired_bulk );
```

Infoblox::Session->search( )

Use this method to search for Bulk Host objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
    Apply the following attributes to search for a specific Bulk Host object:

     zone       - Optional. Zone name in FQDN format (regular expression).
     prefix     - Optional. Prefix of the bulk host (regular expression).
     start_addr - Optional. Start address of the bulk host.
     end_addr   - Optional. End address of the bulk host.
     comment    - Optional. A comment in string format (regular expression).
     view       - Optional. The DNS view in which the bulk host record is located (regular expression).
  The default is all DNS views.

     ttl - Optional. Time to Live (TTL) value in integer format.
     name_template - Optional. The bulk host name template in string format.
                     This is B<not> a regular expression, so do not specify this
                     value to ".*" if you want to obtain BulkHost records with all
                     name_templates. Just leave this field unspecified.
     disable - Optional. Disable flag of a DNS BulkHost record in boolean format.
     use_name_template - Optional. Use name template flag of a DNS BulkHost record in boolean format.
     reverse - Optional. Reverse flag of a DNS BulkHost record in boolean format.
```

Example

```
    # search for all Bulk Host objects that start with "prefix"
    my @retrieved_objs = $session->search(
        object => "Infoblox::DNS::BulkHost",
        prefix => "prefix.*"
    );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a BulkHost object.

comment( )

Use this method to set or retrieve the descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
    #Get comment
    my $comment = $bulkhost->comment();
    #Modify comment
    $bulkhost->comment("Modifying the BulkHost comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS BulkHost record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS BulkHost record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $bulkhost->disable();
#Modify disable
$bulkhost->disable("true");
```

end_addr( )

Use this method to set or retrieve the last IP address in the address range of the BulkHost.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IP address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get end_addr
my $end_addr = $bulkhost->end_addr();
#Modify end_addr
$bulkhost->end_addr("1.0.0.6");
```

name_template( )

Use this method to set or retrieve the bulk host name template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates the grid level default bulk host name format is used.

Specify a bulk host name format value to override the name format at the grid level.

Parameter

The name of a bulk host name format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name_template
my $name_template = $bulkhost->name_template();
#Modify name_template
$bulkhost->name_template("Three Octets");
```

prefix( )

Use this method to set or retrieve the prefix for the BulkHost. The prefix is the name (or a series of characters) inserted at the beginning of each host name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired text with the prefix for the BulkHost. The sum of the bulk host prefix length and suffix length must not exceed 63 characters. If it does, the appliance displays an error message indicating the number of characters that you must remove to

make a valid prefix.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get prefix
my $prefix = $bulkhost->prefix();
#Modify prefix
$bulkhost->prefix("test");
```

reverse( )

Use this method to set or retrieve the reverse flag of the BulkHost record.

If the flag is enabled, reverse mapping `address(es)` are added automatically if a reverse mapping zone exists.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get reverse
my $reverse = $bulkhost->reverse();
#Modify reverse
$bulkhost->reverse("true");
```

start_addr( )

Use this method to set or retrieve the first IP address in the address range of the BulkHost.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IP address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get start_addr
my $start_addr = $bulkhost->start_addr();
#Modify start_addr
$bulkhost->start_addr("1.0.0.2");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $bulkhost->ttl();
#Modify ttl
$bulkhost->ttl(3600);
```

use_name_template( )

Use this method to set or retrieve the use_name_template flag value. This setting overrides the grid level bulk host name format.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get use_name_template
my $use_name_template = $bulkhost->use_name_template();
#Modify use_name_template
$bulkhost->use_name_template("true");
```

views( )

Use this method to set or retrieve the view of a BulkHost.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" view, which means the BulkHost object is located under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $views = $bulkhost->views();
#Modify views
$bulkhost->views([$view1]);
```

zone( )

Use this method to set or retrieve the zone name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A zone name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host_name followed by the domain name (example: abc.com). A host_name can have a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get zone
my $zone = $bulkhost->zone();
#Modify zone
$bulkhost->zone("test.com");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a Bulk Host object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the zone prior to a BulkHost insertion
my $zone = Infoblox::DNS::Zone->new(
    name    => "domain.com",
    email   => "admin\@domain.com",
    comment => "add a zone domain.com",
    );
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a BulkHost object

```
my $bulk_host = Infoblox::DNS::BulkHost->new(
    zone               => "domain.com",
    comment            => "add bulk host",
    prefix             => "prefix",
    start_addr         => "1.0.0.1",
    end_addr           => "1.0.0.5",
    use_name_template  => "true",
    name_template      => "Three Octets",
    reverse            => "true"
    );
unless($bulk_host) {
     die("Construct BulkHost failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "BulkHost object created successfully\n";

#Add the BulkHost object to Infoblox Appliance through a session
$session->add($bulk_host)
    or die("Add BulkHost object failed: ",
           $session   status code            $session   status detail
```

```
                    ->            () . ":" .          ->             ());
print "Bulk Host object added to server successfully\n";
```

#Search for a specific DNS BulkHost record

```
#Search for all BulkHost objects in the zone "domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::BulkHost",
    zone   => "domain.com"
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search Bulk Host object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search Bulk Host object found at least 1 matching entry\n";

#Search for all BulkHost objects that start with prefix as "prefix"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::BulkHost",
    prefix => "prefix.*"
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search Bulk Host object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search Bulk Host object found at least 1 matching entry\n";

#Search for all BulkHost objects with specific start_addr
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::BulkHost",
    start_addr => "1.0.0.1",
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search Bulk Host object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search Bulk Host object found at least 1 matching entry\n";

#Search for all BulkHost objects with specific end_addr
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::BulkHost",
    end_addr => "1.0.0.5",
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search Bulk Host object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search Bulk Host object found at least 1 matching entry\n";
```

#Get and modify a BulkHost object

```
#Get BulkHost object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::BulkHost",
    zone   => "domain.com",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get BulkHost object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get BulkHost object found at least 1 matching entry\n";

#Modify one of the attributes of the specified BulkHost object
$object->prefix("test");

#Apply the changes
$session->modify($object)
    or die("Modify BulkHost object failed: ",
```

```
            $session->status_code() . ":" . $session->status_detail());
print "BulkHost object modified successfully \n";
```

#Remove a BulkHost object

```
#Get BulkHost object through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::DNS::BulkHost",
    zone       => "domain.com",
    prefix     => "test",
    start_addr => "1.0.0.1",
    end_addr   => "1.0.0.5"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get BulkHost object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get BulkHost object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove  BulkHost object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "BulkHost object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::DnssecTrustedKey - DNSSEC trusted key object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DNS->dnssec_trusted_keys( )
  - Infoblox::Grid::Member::DNS->dnssec_trusted_keys( )
  - Infoblox::DNS::View->dnssec_trusted_keys( )
- METHODS
  - algorithm( )
  - fqdn( )
  - key( )
  - secure_entry_point( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::DnssecTrustedKey - DNSSEC trusted key object.

DESCRIPTION

For recursively derived signed data, the DNS server can perform validation. In order to be secure, this validation relies on a set of trust anchors. These trust anchors and their associated trusted keys can be specified via the Infoblox::DNS::DnssecTrustedKey objects.

DNSSEC is based on DNS specifications RFC 4033, RFC 4034, and RFC 4035, which provide authentication protocol enhancements to secure DNS transactions.

CONSTRUCTOR

```
 my $dnssec_trusted_key=Infoblox::DNS::DnssecTrustedKey(
                    algorithm              =>
"RSAMD5"|"DSA"|"RSASHA1"|"NSEC3DSA"|"NSEC3RSASHA1"|"RSASHA256"|"RSASHA512"|"NSEC3RSASHA256"|"NSEC3RSASHA512"|1|3|5|6|
,  #Required
                    fqdn                   => $string,                                  #Required
                    key                    => $string,                                  #Required
                    secure_entry_point      => "true"|"false"                           #Optional / Default is "true"
                        );
```

MODULE METHODS

This section describes all the functions that you can apply to a DnssecTrustedKey object.

Infoblox::Grid::DNS->dnssec_trusted_keys( )

Use this function to specify a list of DNSSEC keys for a grid object.

Example

```
 ## Construct the trusted key object. When trying to implement this example replace with the actual public key
of the signed zone.
 my $dnssec_key = Infoblox::DNS::DnssecTrustedKey->new(
     algorithm => "RSAMD5",
     fqdn => "example.com",
     key  => "Zx1jVJl7C58GT/sc7Q3ucA==",
     secure_entry_point => "false"
 );
 # Configure dnssec_trusted_keys on the grid DNS object
 my $response = $Grid_DNS->dnssec_trusted_keys([$dnssec_key]);
```

Infoblox::Grid::Member::DNS->dnssec_trusted_keys( )

Use this function to specify a list of DNSSEC keys for a grid member object.

Example

```
 ## Construct the trusted key object
 my $dnssec_key = Infoblox::DNS::DnssecTrustedKey->new(
     algorithm => "RSAMD5",
     fqdn => "example.com",
     key  => "Zx1jVJl7C58GT/sc7Q3ucA==",
     secure_entry_point => "false"
 );
 # Configure dnssec_trusted_keys on the grid DNS object
```

```
my $response = $Grid_Member_DNS->dnssec_trusted_keys([$dnssec_key]);
```

Infoblox::DNS::View->dnssec_trusted_keys( )

Use this function to specify a list of DNSSEC keys for a DNS View object.

Example

```
## Construct the trusted key object
my $dnssec_key = Infoblox::DNS::DnssecTrustedKey->new(
    algorithm => "RSAMD5",
    fqdn => "example.com",
    key  => "Zx1jVJl7C58GT/sc7Q3ucA==",
    secure_entry_point => "false"
);
# Configure dnssec_trusted_keys on the grid DNS object
my $response = $DNS_View->dnssec_trusted_keys([$dnssec_key]);
```

METHODS

algorithm( )

Use this method to set or retrieve a DNSSEC encryption algorithm.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The public-key encryption algorithm. This can be either an integer or mnemonic. The following table shows the correlation between the integers and mnemonics for the algorithms supported by the appliance:

Value	Algorithm	Mnemonic
1	RSA/MD5	RSAMD5
3	DSA	DSA
5	RSA/SHA-1	RSASHA1
6	NSEC3DSA	NSEC3DSA
7	NSEC3RSA/SHA-1	NSEC3RSASHA1

For algorithms not listed above, only the mnemonic representation is supported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting algorithm
my $algorithm = $dnssec_key->algorithm();
# Modifying algorithm
$dnssec_key->algorithm("DSA");
```

fqdn( )

Use this method to set or retrieve the FQDN of the zone to which the key belongs.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

FQDN of the zone to which the trusted_key pertains, in DNS label order.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting fqdn
my $zone = $dnssec_key->fqdn();
# Modifying algorithm
$dnssec_key->fqdn("zone.example.com");
```

key( )

Use this method to add or retrieve the DNSSEC public key for the zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Base-64 encoding of the key.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting key
my $key = $dnssec_key->key();
# Modifying key
$dnssec_key->key("AQPdWbrGbVv1eDhNgRhpJMPonJfA3reyEo82ekwRnjbX7+uBxB11BqL7LAB7/C+eb0vCtI53FwMhkkNkTmA6bI8B");
```

secure_entry_point()

Use this method to set or retrieve the secure entry point value for the field.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set to "true" to enable the secure entry point, set to "false" to disable it. Default value of this parameter is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting secure_entry_point
my $sep = $dnssec_key->secure_entry_point();
# Modifying secure_entry_point
$dnssec_key->secure_entry_point("false");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add and modify. This sample also includes error handling for the operations.

##Preparation prior to a DnssecTrustedKey object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox  appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DnssecTrustedKey object

```
 my $dnssec_key = Infoblox::DNS::DnssecTrustedKey->new(
    key  => "ZX1jVJl7C58GT/sc7Q3ucA==",
    fqdn => "myzone.example.com",
    algorithm => "RSASHA1"
  );

unless($dnssec_key) {
     die("Construct DnssecTrustedKey failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DnssecTrustedKey object created successfully\n";
```

#Get the member DNS object and add a DnssecTrustedKey object to it

```
my @result_array = $session->get(
    object => "Infoblox::Grid::Member::DNS",
    name   => "infoblox.localdomain"
);

unless (scalar(@result_array) == 0) {
    my $memberdns = $result_array[0];

    if ($memberdns) {
```

```
        #Apply changes to the Member object.
                $memberdns->dnssec_trusted_keys([$dnssec_key])
            or die("modify member failed: ",
                    Infoblox::status_code() . ":" . Infoblox::status_detail());
        print "DnssecTrustedKey added to Member DNS object successfully\n";

  #Update member DNS object through the Infoblox session.
        $session->modify($memberdns)
            or die("modify session failed: ",
                    $session->status_code() . ":" . $session->status_detail());
        print "Member DNS object with DnssecTrustedKey updated to Infoblox device successfully\n";

         #Modify existing DnssecTrustedKey object

        #Modifying the value of the specified object.
        $dnssec_key->fqdn("another.example.com");
        print "Modify DnssecTrustedKey fqdn value\n";


        #Apply changes to the member object.
        $memberdns->transfer_keys([$dnssec_key])
            or die("modify member failed: ",
                    Infoblox::status_code() . ":" . Infoblox::status_detail());
        print "DnssecTrustedKey fqdn updated to Member DNS object successfully\n";

        #Update member DNS object through the Infoblox session.
        $session->modify($memberdns)
            or die("modify session failed: ",
                $session->status_code() . ":" . $session->status_detail());
                print "Member DNS object with DnssecTrustedKey fqdn updated to Infoblox device successfully\n";
             }
         } else {
             print "No member found with the specified name.";
         }

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member::DNS, Infoblox::DNS::View, Infoblox::Grid::DNS

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::GlueRecordAddr - Infoblox DNS Glue Record Object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::Member::DNS->glue_record_addresses( )
- METHODS
  - attach_empty_recursive_view( )
  - glue_record_address( )
  - view( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::GlueRecordAddr - Infoblox DNS Glue Record Object.

DESCRIPTION

The GlueRecordAddr allows you to specify which interface address the appliance uses to generate the glue record for each view.

CONSTRUCTOR

The Infoblox::DNS::GlueRecordAddr object is automatically generated upon the successful creation of the Infoblox::DNS::View object. It does not require manual construction.

MODULE METHODS

This section describes all the functions that you can apply to a GlueRecordAddr object.

Infoblox::Grid::Member::DNS->glue_record_addresses( )

Use this function to specify a glue record address for a view at the grid member level. See Infoblox::Grid::Member::DNS->glue_record_addresses() for parameters and return values.

Example

```
# Retrieve the list of GlueRecordAddr objects from Member DNS object.
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member::DNS",
    name   => "infoblox.localdomain" );

#select the desired GlueRecordAddr for the view
my $Grid_Member_DNS = $retrieved_objs[0];
my $ref_all_glue_record_addr = $Grid_Member_DNS->glue_record_addresses();
my $glue_record_addr = @$ref_all_glue_record_addr[0];

# Modify Glue Record Address for the view to use NAT address.
$glue_record_addr->glue_record_address("NAT");

#Save modification to Member DNS object.
my $response = $Grid_Member_DNS->glue_record_addresses($ref_all_glue_record_addr);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a GlueRecordAddr object.

attach_empty_recursive_view( )

Use this method to set or retrieve the attach_empty_recursive_view flag of glue record object. If the attach_empty_recursive_view flag is set to "true", then empty view with recursion enabled will be written into the conf file.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the attach_empty_recursive_view flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get attach_empty_recursive_view.
my $attach_empty_recursive_view = $glue_record_addr->attach_empty_recursive_view();
#Modify attach_empty_recursive_view.
$glue_record_addr->attach_empty_recursive_view("true");
```

glue_record_address( )

Use this method to set or retrieve the address the appliance uses to generate the glue record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

String which specifies the interface address to use. Valid values are "interface", "NAT", or specify an IPv4 address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get glue record address.
my $glue_record_address = $glue_record_addr->glue_record_address();
#Modify glue record address.
$glue_record_addr->glue_record_address("NAT");
```

view( )

Use this method to retrieve the view of this Glue Record object

Parameter

None.

Returns

The method returns the attribute value.

Example

```
#Get view
my $glue_record_view = $glue_record_addr->view();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as get and modify. This sample also includes error handling for the operations.

#Preparation prior to a DNS GlueRecordAddr object modification

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
          master   => "192.168.1.2",
          username => "admin",
          password => "infoblox"
);

unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
```

```
print "Session created successfully\n";

# Retrieve the list of GlueRecordAddr objects from Member DNS object.
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member::DNS",
    name   => "infoblox.localdomain" );

my $Grid_Member_DNS = $retrieved_objs[0];
unless ($Grid_Member_DNS) {
   die("Retrieve Grid Member DNS object failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Grid Member DNS object retrieve successfully\n";
```

#Modify the Glue Record address

```
my $ref_all_glue_record_addr = $Grid_Member_DNS->glue_record_addresses();

# Select the desired GlueRecordAddr for the view
my $glue_record_addr = @$ref_all_glue_record_addr[0];
unless ($glue_record_addr) {
   die("Glue Record Address not found on Grid Member DNS object: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Glue Record Address found on Grid Member DNS object\n";

# Modify Glue Record Address for the view to use a custom defined address.
$glue_record_addr->glue_record_address("interface")
    or die("modify glue record failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Glue Record Address modified successfully\n";

# Save modification to Member DNS object.
my $response = $Grid_Member_DNS->glue_record_addresses($ref_all_glue_record_addr);
unless($response) {
    die("Modify glue record on Member DNS failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Glue Record Address updated to Grid Member DNS object successfully\n";

#submit changes to Infoblox appliance
$session->modify($Grid_Member_DNS)
   or die("Modify Member DNS object on Infoblox appliance failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Grid Member DNS modified on Infoblox appliance successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session->get(), Infoblox::Session->modify(),Infoblox::Grid::Member::DNS->glue_record_addresses()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Host - DNS Host record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - aliases( )
  - comment( )
  - configure_for_dns( )
  - disable( )
  - extensible_attributes( )
  - ipv4addrs( )
  - ipv6addrs( )
  - name( )
  - network_view( )
  - rrset_order( )
  - ttl( )
  - views( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Host - DNS Host record object.

DESCRIPTION

A host record defines attributes for a node, such as the name-to-address and address-to-name mapping. This alleviates having to specify an A record and a PTR record separately for the same node. A host can also define aliases and DHCP fixed address nodes. The zone must be created first before adding a host record for the zone.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
 my $host = Infoblox::DNS::Host->new(
     ipv4addrs    => [$ipv4addr, $ipv4addr, $FixedAddr, ...], #Required
     name         => $fqdn,                                   #Required
     aliases      => [$fqdn, $fqdn, ...],                      #Optional / Default is empty
     comment      => $string,                                 #Optional / Default is undefined
     configure_for_dns      => "true" | "false",               #Optional / Default "true"
     disable      => "true" | "false",                        #Optional / Default "false"
     extensible_attributes        => { $string => $string | $num, $string => [ $string | $num, ... ], ...
} #Optional / Default is undefined
     ipv6addrs    => [$ipv6addr, $ipv6addr, ...],   #Optional / Default is empty
     network_view => $NetworkView,                            #Optional / Default is "default" network view
     rrset_order  => "cyclic" | "random" | "fixed",   #Optional / Default "cyclic"
     ttl          => $num | undef,                            #Optional / Default is undefined
     views        => [$View]                                    #Optional / Default is "default" view
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a DNS host record object.

Infoblox::Session->add( )

Use this method to add a host record object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $host = Infoblox::DNS::Host->new(
    name         => "host1.test.com",
    aliases      => ["one.foo.com", "two.foo.com"],
    comment      => "add a host host1.test.com",
    disable      => "true",
    ipv4addrs    => ["4.4.4.4"],
    ipv6addrs    => ["2001::1:2:3:4"],
    newtork_view => $nview,
    ttl          => 3600,
    views        => [$view1],
);

# Submit for addition
my $response = $session->add( $host );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS host object:

  name          - Optional. A host name in string format.
  network_view  - Optional. The network view in which the host record is located. By default, all
network views are searched.
  view          - Optional. The DNS view in which the host record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
  zone          - Optional. A zone name in FQDN format.
  comment       - Optional. A comment in string format (regular expression)
  mac           - Optional. A MAC address present in any FixedAddr object in the ipv4addrs member.
  ipv4addr      - Optional. An IP address in string format (regular expression).
  ipv6addr      - Optional. An IPv6 address in string format (regular expression).
  alias         - Optional. An alias of the host in string format (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
my @retrieved_objs = $session->get(
                     object       => "Infoblox::DNS::Host",
                     name         => "host1.test.com",
                     network_view => "default",
                     view         => "default" );

my @retrieved_objs = $session->get(
                     object                => "Infoblox::DNS::Host",
                     extensible_attributes => { 'Site' => 'Santa Clara' });

# get all DNS host objects in zone "test.com" of default view
my @retrieved_objs = $session->get(
                     object       => "Infoblox::DNS::Host",
                     zone         => "test.com",
                     network_view => "default",
                     view         => "default" );
```

Infoblox::Session->modify( )

Use this method to modify a host record object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$host->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $host );
```

Infoblox::Session->remove( )

Use this method to remove a host record object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific host record object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Host",
    name   => "host1.test.com",
    view   => "default" );
# Find the desired object from the retrieved list.
my $desired_host = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_host );
```

Infoblox::Session->search( )

Use this method to search for DNS host record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes, to search for a DNS host record object:

  name          - Optional. A host name in string format (regular expression).
  network_view  - Optional. The network view in which the host record is located. By default, all
network views are searched.
  view          - Optional. The DNS view in which the host record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
  zone          - Optional. A zone name in FQDN format.
  comment       - Optional. A comment in string format (regular expression).
  mac           - Optional. A MAC address present in any FixedAddr object in the ipv4addrs member
(regular expression).
  ipv4addr      - Optional. An IP address in string format (regular expression).
  ipv6addr      - Optional. An IPv6 address in string format (regular expression).
  alias         - Optional. An alias of the host in string format (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
# search for all DNS host objects that match ".test.com" in the default DNS view
my @retrieved_objs = $session->search(
    object  => "Infoblox::DNS::Host",
    name    => '.*\.test\.com',
    view    => "default",
    comment => ".*modified comment" );

# search for DNS host objects defining "Santa Clara" for "Site" extensible attribute
my @retrieved_objs = $session->search(
    object                => "Infoblox::DNS::Host",
    extensible_attributes => { 'Site' => 'Santa Clara' });

# search for all DNS host objects in zone "test.com" of default view
my @retrieved_objs = $session->search(
    object       => "Infoblox::DNS::Host",
    zone         => "test.com",
    network_view => "default",
    view         => "default" );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a host record.

aliases( )

Use this method to set or retrieve the aliases of the host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference with the aliases for the host. The aliases of the host should be in Fully Qualified Domain Name (FQDN) format. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting aliases
my $ref_aliases = $host->aliases();
#Modifying aliases
$host->aliases(["one1.foo.com", "two2.foo.com"]);
```

comment( )

Use this method to set or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $host->comment();
#Modify comment
$host->comment("Modifying the DNS host comment");
```

configure_for_dns( )

Use this method to set or retrieve the configure_for_dns flag of a DNS host.

When configure_for_dns is false the host will not have parent zone information.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is true.

Parameter

Specify "true" to set the configure_for_dns flag or "false" to deactivate/unset it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get configure_for_dns
my $configure_for_dns = $host->configure_for_dns();
#Modify configure_for_dns
$host->configure_for_dns("false");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS host.

The host can be disabled instead of removed. You can avoid removing and re-adding a host when a network appliance is repaired or relocated. When the changes to the physical appliance are complete, the host can be simply re-enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
```

```
my $disable = $host->disable();
#Modify disable
$host->disable("false");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS Host object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = host->extensible_attributes();
#Modify extensible attributes
$host->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

ipv4addrs( )

Use this method to set or retrieve the IPv4 addresses of the host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains defined Infoblox::DHCP::FixedAddr objects or IPv4 addresses.

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192. 168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Gett ipv4addrs
my $ref_ipv4addr = $host->ipv4addrs();
#Modify ipv4addrs
$host->ipv4addrs([$fixedaddr, "2.2.2.2"]); #$fixedaddr is an Infoblox::DHCP::FixedAddr object
```

ipv6addrs( )

Use this method to set or retrieve the IPv6 addresses of the host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv6 addresses in string format.

An IPv6 address is a string consisting of eight groups of four hexadecimal digits, where each group is separated by a colon (:) (example: 2001:0db8:85a3:0000:0000:8a2e:0370:7334). Various shortcuts exist to shorten the string representation of an IPv6 address, such as omitting the leading zeros of each group and replacing one or any number of consecutive groups of 0 value with two colons(::). i.e. The previous example can be written as 2001:db8:85a3::8a2e:370:7334.

When creating an Infoblox::DNS::Host object, setting this method to [] is not supported. If the Infoblox::DNS::Host object does not have associated IPv6 addresses, simply do not pass the ipv6addrs attribute to the constructor. If [] is set and the Host object is submitted for addition, the following error will be reported: "A host record requires at least one IP address." Setting the method to [] is supported for modify requests.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Gett ipv6addrs
my $ref_ipv6addr = $host->ipv6addrs();
#Modify ipv6addrs
$host->ipv6addrs(["2001::1", "2002::1:2:3:4"]);
```

name( )

Use this method to set or retrieve the host name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Host name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A host name can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $host->name();
#Modify name
$host-> name("host2.test.com");
```

network_view( )

Use this method to set or retrieve the network view of the DNS host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" network view, which means the DNS host is in the default network view.

Parameter

Valid value is a defined Infoblox::DHCP::View object. The default network view value is the system-defined default network view object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $host->network_view();
#Modify network view, with an Infoblox::DHCP::View object
$host->network_view($nview);
```

rrset_order( )

Use this method to set or retrieve the rrset_order value. The value of this parameters specifies the order in which resource record sets are returned.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The possible values of the parameter are "cyclic", "random" and "fixed". The default value of this parameter is "cyclic".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get rrset_order
my $order = $host->rrset_order()
# Set fixed RR set order
$host->rrset_order("fixed");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $host->ttl();
#Modify ttl
$host->ttl(1800);
#Un-override ttl
$host->ttl(undef);
```

views( )

Use this method to set or retrieve the view of the DNS host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" view, which means the DNS host is located under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects. The default view value is "default".

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $host->views();
#Modify views, list of Infoblox::DNS::View objects
$host->views([$view1]);
```

zone( )

Use this method to retrieve the zone name of a DNS host object. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
```

```
my $zone = $host->zone();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a Host record object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS host record insertion

```
#PROGRAM STARTS: Include all the modules that will be used

use strict;
use Infoblox;

#Create a session to the Infoblox Appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2", #appliance host ip
    username => "admin",     #appliance user login
    password => "infoblox"   #appliance password
);

unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

#Create the zone prior to an host insertion
my $zone = Infoblox::DNS::Zone->new(name => "test.com");

unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone",
                           name => "test.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS host record

```
#Construct a DNS host object

my $host = Infoblox::DNS::Host->new(
    name         => "host1.test.com",
    aliases      => ["one.foo.com", "two.foo.com"],
    comment      => "add a host host1.test.com",
    disable      => "true",
    ipv4addrs    => ["4.4.4.4"],
    ipv6addrs    => ["2001::4:4:4:4"],
    ttl          => 3600,
);

unless ($host) {
   die("Construct DNS host record failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS host object created successfully\n";

#Add the DNS host record object to Infoblox appliance through a session
$session->add($host)

    or die("Add host record failed: ",
           $session->status_code() . ":" . $session->status_detail());
     print "DNS host object added to server successfully\n";
```

#Search for a specific DNS host record

```
#Search host record host1.test.com
```

```
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Host",
    name   => 'host1\.test\.com'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search host record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS host object found at least 1 matching entry\n";

#Search all host records that start with "add" in the comment
my @retrieved_objs = $session->search(
    object  => "Infoblox::DNS::Host",
    comment => "add.*",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search host record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS host object using regexp in comment field found at least 1 matching entry\n";

#Search all host records that start with "host1" in the zone
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Host",
    name   => 'host1.*\.com',
 );
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search host record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS host object using regexp in name field found at least 1 matching entry\n";
```

#Get and modify a DNS host record

```
#Get host record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Host",
    name   => "host1.test.com"
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get host record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

#Modify one of the attributes of the specified host record
$object->comment("This is the modified comment");

#Applying the changes
$session->modify($object)
    or die("Modify host record failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS host object modified successfully \n";
```

#Remove a DNS host record.

```
#Get host record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Host",
    name   => "host1.test.com"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get host record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Get DNS host object successfull\n";

#Submit the object for removal

$session->remove($object)
    or die("Remove host record failed: ",
```

```
        $session->status_code() . ":" . $session->status_detail());
print "DNS host object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DHCP::View, Infoblox::DNS::Zone, Infoblox::DHCP::FixedAddr,Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Member - DNS Member object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DNS::Zone->primary( )
  - Infoblox::DNS::Zone->secondary( )
  - Infoblox::Grid::DNS::Nsgroup->primary( )
  - Infoblox::Grid::DNS::Nsgroup->secondary( )
- METHODS
  - grid_replicate( )
  - ipv4addr( )
  - ipv6addr( )
  - lead( )
  - name( )
  - stealth( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Member - DNS Member object

DESCRIPTION

A grid member is part of a grid which is a group of two or more Infoblox appliances that share sections of a common, distributed, built-in database. A grid member inherits its settings from the grid. A member is a single Infoblox appliance or an HA pair that provides DNS services seamlessly across an entire network.

CONSTRUCTOR

```
my $member = Infoblox::DNS::Member->new(
    ipv4addr       => $ipv4addr,        #Required if $ipv6addr is not present
    ipv6addr       => $ipv6addr,        #Required if $ipv4addr is not present
    grid_replicate => "true" | "false", #Optional / Default is "false"
    lead           => "true" | "false", #Optional / Default is "false"
    name           => $fqdn,            #Optional / Default is  hostname of the matching ipv4addr
    stealth        => "true" | "false"  #Optional / Default is "false"
);
```

MODULE METHODS

The following functions are available to apply to an DNS Member object.

Infoblox::DNS::Zone->primary( )

Use this function to specify primary member to a zone. See Infoblox::DNS::Zone->primary() for parameters and return values.

Example

```
my $member1 = Infoblox::DNS::Member->new(
    name           => "localhost.localdomain",
    ipv4addr       => "192.168.1.2",
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);

unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create the Zone object with this member
```

```
my $firstzone = Infoblox::DNS::Zone->new(
    name        => "infoblox.com",
    primary     => $member1,
);

my $response = $session->add($firstzone)
unless($response) {
     die("Add zone failed: ",
           session->status_code() . ":" . session->status_detail());
}
print "DNS member object added to the zone successfully\n";
```

Infoblox::DNS::Zone->secondary( )

Use this function to specify secondary member to a zone. See Infoblox::DNS::Zone->secondary() for parameters and return values.

Example

```
my $member1 = Infoblox::DNS::Member->new(
    name           => "localhost.localdomain",
    ipv4addr       => "192.168.1.2",
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);

unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

my $member2 = Infoblox::DNS::Member->new(
    name           => "localhost1.localdomain",
    ipv4addr       => "192.168.1.3",
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);

unless($member2) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create the Zone object with this member
my $firstzone = Infoblox::DNS::Zone->new(
    name        => "infoblox.com",
    primary     => $member1,
    secondaries => [ $member2 ] ,
);

my $response = $session->add($firstzone)
unless($response) {
     die("Add zone failed: ",
           session->status_code() . ":" . session->status_detail());
}

print "DNS member object added to the zone successfully\n";
```

Infoblox::Grid::DNS::Nsgroup->primary( )

Use this function to specify primary member to a NS group. See Infoblox::Grid::DNS::Nsgroup->primary() for parameters and return values.

Example

```
my $member1 = Infoblox::DNS::Member->new(
    name           => "localhost.localdomain",
    ipv4addr       => "192.168.1.2",
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);
```

```
unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

my $nsgroup1 = Infoblox::Grid::DNS::Nsgroup->new (
   name      => "ns_group1",
   primary   => $member1
);
```

Infoblox::Grid::DNS::Nsgroup->secondary( )

Use this function to specify secondary member to a NS group. See Infoblox::Grid::DNS::Nsgroup->secondary() for parameters and return values.

Example

```
my $member1 = Infoblox::DNS::Member->new(
    name           => "localhost.localdomain",
    ipv4addr       => "192.168.1.2",
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);

unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

my $member2 = Infoblox::DNS::Member->new(
    name           => "localhost1.localdomain",
    ipv4addr       => "192.168.1.3",
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);

unless($member2) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

my $nsgroup1 = Infoblox::Grid::DNS::Nsgroup->new (
   name        => "ns_group1",
   primary     => $member1
   secondaries => [ $member2 ],
);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a DNS Member object.

grid_replicate( )

Use this method to set or retrieve the grid_replicate flag, which indicates whether grid members that are primary and secondary servers use database replication or zone transfers for zone updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to use ID grid replication or "false" to use DNS zone transfers.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get grid replicate
my $grid_replicate = $member->grid_replicate();
#Modify grid replicate
$member->grid_replicate("true");
```

ipv4addr( )

Use this method to set or retrieve the IPv4 address of the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

IPv4 address (32 bits) of the member.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4Addr
my $ipv4Addr = $member->ipv4Addr();
#Modify ipv4Addr
$member->ipv4Addr("192.168.1.3");
```

ipv6addr( )

Use this method to set or retrieve the IPv6 address if available. Required only when ipv4 is not specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv6 address is a 128-bit number in colon hexadecimal notation. It consists of eight groups of four hexadecimal digits separated by colons (example: 12ab:0000:0000:0123:4567:89ab:0000:cdef).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting ipv6addr
my $ipv6addr = $member->ipv6addr();
# Modifying ipv6addr
$member->ipv6addr("12ab::345");
```

lead( )

Use this method to set or retrieve the "lead" flag.

When a primary server is external to a grid whose members are secondary servers, you can designate one member as a lead secondary. The primary server sends zone transfers to the lead secondary, which distributes the zone data to the other secondary servers in the grid using zone transfers (not the ID grid data replication mechanism).

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set lead flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lead
my $lead = $member->lead();
#Modify lead
$member->lead("true");
```

name( )

Use this method to set or retrieve the name of the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: host.domain.com). A host name can have a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $member->name();
#Modify name
$member->name("dns.infoblox.com");
```

stealth( )

Use this method to set or retrieve the "stealth" flag, which indicates whether this member is in stealth mode. When a member is in stealth mode, its NS record is hidden from DNS queries; it does not respond to queries from resolvers and other name servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the stealth flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get stealth
my $stealth = $member->stealth();
#Modify stealth
$member->stealth("true");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS Member object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $host_ip   = "192.168.1.2";
my $host_name = "infoblox.localdomain";

my $session = Infoblox::Session->new(
               master   => $host_ip, #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DNS Member object

```
my $member1 = Infoblox::DNS::Member->new(
    name           => $host_name,
    ipv4addr       => $host_ip,
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);

unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

my $default_view = Infoblox::DNS::View->new(name => "default");

#Create the Zone object with this member
my $firstzone = Infoblox::DNS::Zone->new(
    name        => "infoblox.com",
    views       => [$default_view],
    email       => "admin\@infoblox.com",
    comment     => "add a zone infoblox.com",
    primary     => $member1,
);

unless($firstzone) {
            die("Construct Zone object failed: ",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Add the Zone object into the Infoblox appliance through a session
$session->add($firstzone)
     or die("Add Zone object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS member object added to the zone successfully\n";

#Modify a DNS member

my @result = $session->search(object => "Infoblox::DNS::Zone",
                              name   => "infoblox.com");

 unless (scalar(@result) == 0) {
     $firstzone = $result[0];

     if ($firstzone) {

         #modify grid replicate value of member
         $member1->grid_replicate("true");

         #update zone
         $firstzone->primary($member1);

         #update session
         $session->modify($firstzone)
                or die("Modify zone object failed: ",
                      $session->status_code() . ":" . $session->status_detail());
         print "DNS member object updated and added to the zone successfully\n";
               }
           } else {
               print "No zone found with the specified name";
           }

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::TSIGKey, Infoblox::Session->add(),Infoblox::Session->modify(),Infoblox::Session->search(),Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::MSServer - Microsoft DNS Server object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DNS::Zone->primary( )
  - Infoblox::DNS::Zone->secondaries( )
  - Infoblox::DNS::Zone->stub_members( )
  - Infoblox::Grid::DNS::Nsgroup->primary( )
  - Infoblox::Grid::DNS::Nsgroup->secondaries( )
- METHODS
  - address( )
  - ipv4addr( )
  - name( )
  - stealth( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::MSServer - Microsoft (r) DNS Server object

DESCRIPTION

A Microsoft DNS Server object is a Windows-based server that is running the DNS service and is synchronized with an Infoblox grid. As such, the Microsoft DNS server shares its DNS data with the Infoblox DNS servers in a grid.

CONSTRUCTOR

```
my $msserver = Infoblox::DNS::MSServer->new(
    address       => $string,           #Required
    ipv4addr      => $string,           #Required
    name          => $string,           #Required
    stealth       => "true" | "false",  #Optional
);
```

MODULE METHODS

The following functions can be applied to a Microsoft DNS Server object.

Infoblox::DNS::Zone->primary( )

Use any of these functions to assign a Microsoft DNS server as the primary server of a zone. See Infoblox::DNS::Zone->primary() for parameters and return values.

Example

```
my $msserver1 = Infoblox::DNS::MSServer->new(
    address        => "10.0.0.1",
    ipv4addr       => "10.0.0.1",
    name           => "hostname.com",
);

unless($msserver1) {
     die("Construct msserver failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Msserver object created successfully\n";

#Create the Zone object with this Microsoft DNS server
my $firstzone = Infoblox::DNS::Zone->new(
    name        => "infoblox.com",
    primary     => [$msserver1],
);

my $response = $session->add($firstzone)
unless $response
```

```
        (          ) {
     die("Add zone failed: ",
           session->status_code() . ":" . session->status_detail());
}
print "DNS MS Server object added to the zone successfully\n";
```

Infoblox::DNS::Zone->secondaries( )

Use this function to specify an Microsoft DNS server as the secondary server to a zone. See Infoblox::DNS::Zone->secondaries() for parameters and return values.

Example

```
my $msserver1 = Infoblox::DNS::MSServer->new(
    address        => "10.0.0.1",
    ipv4addr       => "10.0.0.1",
    name           => "hostname.com",
);

unless($msserver1) {
     die("Construct msserver failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Msserver object created successfully\n";

#Create the Zone object with this member
my $firstzone = Infoblox::DNS::Zone->new(
    name        => "infoblox.com",
    primary     => $member1,
    secondaries => [ $msserver2 ] ,
);

my $response = $session->add($firstzone)
unless($response) {
     die("Add zone failed: ",
           session->status_code() . ":" . session->status_detail());
}

print "DNS MS server member object added to the zone successfully\n";
```

Infoblox::DNS::Zone->stub_members( )

Use this function to assign a Microsoft DNS server to a stub zone. See Infoblox::DNS::Zone->stub_members() for parameters and return values.

Example

```
my $msserver1 = Infoblox::DNS::MSServer->new(
    address        => "10.0.0.1",
    ipv4addr       => "10.0.0.1",
    name           => "hostname.com",
);

unless($msserver1) {
     die("Construct msserver failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Msserver object created successfully\n";

#Create the Zone object with this member
my $firstzone = Infoblox::DNS::Zone->new(
    name        => "infoblox.com",
    stub_members => [ $msserver2 ] ,
);

my $response = $session->add($firstzone)
unless($response) {
     die("Add zone failed: ",
           session->status_code() . ":" . session->status_detail());
}

print "DNS MS server member object added to the zone successfully\n";
```

Infoblox::Grid::DNS::Nsgroup->primary( )

Use this function to add a Microsoft DNS server as the primary server of a Name Server (NS) group. See Infoblox::Grid::DNS::Nsgroup->primary() for parameters and return values.

Example

```
my $msserver1 = Infoblox::DNS::MSServer->new(
    address        => "10.0.0.1",
    ipv4addr       => "10.0.0.1",
    name           => "hostname.com",
);

unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "MS server object created successfully\n";

my $nsgroup1 = Infoblox::Grid::DNS::Nsgroup->new (
   name      => "ns_group1",
   primary   => [$msserver1]
);
```

Infoblox::Grid::DNS::Nsgroup->secondaries( )

Use this function to add secondary servers to an NS group. See Infoblox::Grid::DNS::Nsgroup->secondaries() for parameters and return values.

Example

```
my $msserver1 = Infoblox::DNS::MSServer->new(
    address        => "10.0.0.1",
    ipv4addr       => "10.0.0.1",
    name           => "hostname.com",
);

unless($member1) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "MS Server object created successfully\n";

my $nsgroup1 = Infoblox::Grid::DNS::Nsgroup->new (
   name        => "ns_group1",
   primary     => $member1
   secondaries => [ $msserver1 ],
);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a DNS Member object.

address( )

Use this method to set or retrieve the IPv4 address or FQDN of the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

IPv4 address (32 bits) or FQDN of the member.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get address
my $address = $member->address();
#Modify address
$member->address("192.168.1.3");
```

ipv4addr( )

Use this method to set or retrieve the member IPv4 address that is published in the glue A record when setting this Microsoft DNS server as a name server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

IPv4 address (32 bits) of the member.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $member->ipv4addr();
#Modify ipv4addr
$member->ipv4addr("192.168.1.3");
```

name( )

Use this method to set or retrieve the member FQDN that is used when setting this Microsoft DNS server as a name server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The member's FQDN.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $member->name();
#Modify name
$member->name("hostname.com");
```

stealth( )

Use this method to set or retrieve the "stealth" flag, which indicates whether this name server is in stealth or normal mode.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the stealth flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get stealth
my $stealth = $member->stealth();
#Modify stealth
$member->stealth("true");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::TSIGKey, Infoblox::Session->add(), Infoblox::Session->modify(), Infoblox::Session->search(), Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Nameserver - DNS Nameserver object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DHCP::View->external_ddns_primaries( )
  - Infoblox::DNS::Zone->delegate_to( )
  - Infoblox::DNS::Zone->forward_to( )
  - Infoblox::DNS::Zone->secondaries( )
  - Infoblox::DNS::Zone->primary( )
  - Infoblox::Grid::Member->prefer_resolver( )
  - Infoblox::Grid::Member->alternate_resolver( )
  - Infoblox::Grid::DNS::Nsgroup->primary( )
  - Infoblox::Grid::DNS::Nsgroup->secondaries( )
- METHODS
  - ddns_zone( )
  - gss_tsig_domain( )
  - gss_tsig_dns_principal( )
  - ipv4addr( )
  - ipv6addr( )
  - name( )
  - stealth( )
  - TSIGkey( )
  - TSIGname( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Nameserver - DNS Nameserver object

DESCRIPTION

Manipulates a Nameserver object that contains information about the domain name space.

CONSTRUCTOR

```
 my $nameserver = Infoblox::DNS::Nameserver->new(
     ipv4addr  => $ipv4addr,            #Required if $ipv6addr is not present
     ipv6addr  => $ipv6addr,            #Required if $ipv4addr is not present
     name      => $fqdn,                #Required if this name server is not used to represent an external
DDNS server
     ddns_zone => $string,              #Required if this name server is used to represent an external DDNS
server
     gss_tsig_domain  => $string,       #Optional / Default is undefined
     gss_tsig_dns_principal => $string, #Optional / Default is undefined
     stealth   => "true" | "false",     #Optional / Default is "false"
     TSIGkey   => $string | ":2xCOMPAT" #Optional / Default is undefined
     TSIGname  => $string,              #Optional / Default is undefined
 );
```

MODULE METHODS

The following functions are available to apply to a nameserver object.

Infoblox::DHCP::View->external_ddns_primaries( )

Use this function to specify a nameserver to send DDNS updates to at the network view level.

Example

```
       #Construct an object
         my $nameserver = Infoblox::DNS::Nameserver->new(
           ipv4addr  => "1.2.3.4",
           TSIGname  => "tsig_name",
```

```
    TSIGkey   => "ZX1jVJl7C58GT/sc7Q3uca==",
    ddns_zone => "zone1"
);

#Construct an object
  my $nameserver1 = Infoblox::DNS::Nameserver->new(
    ipv4addr  => "1.2.3.5",
    TSIGname  => "tsig_name",
    TSIGkey   => "ZX1jVJl7C58GT/sc7Q3uca==",
    ddns_zone => "zone2"
);

# Configure Nameserver
my $response = $View->external_ddns_primaries([$nameserver,$nameserver1]);
```

Infoblox::DNS::Zone->delegate_to( )

Use this function to specify a nameserver at the zone level.

Example

```
#Construct an object
  my $nameserver = Infoblox::DNS::Nameserver->new(
    name      => "ns1.test.com",
    ipv4addr  => "4.4.7.7",
);

# Configure Nameserver
my $response = $Zone->delegate_to([$nameserver]);
```

Infoblox::DNS::Zone->forward_to( )

Use this function to specify a nameserver at the zone level.

Example

```
#Construct an object
  my $nameserver = Infoblox::DNS::Nameserver->new(
    name      => "ns2.test.com",
    ipv4addr  => "4.4.6.6",
);

# Configure Nameserver
my $response = $Zone->forward_to([$nameserver]);
```

Infoblox::DNS::Zone->secondaries( )

Use this function to specify a nameserver at the zone level.

Example

```
#Construct an object
  my $nameserver = Infoblox::DNS::Nameserver->new(
    name      => "ns3.test.com",
    ipv4addr  => "4.5.6.4",
    stealth   => "false",
    TSIGname  => "tsig_name",
    TSIGkey   => "ZX1jVJl7C58GT/sc7Q3uca==",
);

#Construct an object
  my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name      => "ns4.test.com",
    ipv6addr  => "2009::546",
    stealth   => "false",
    TSIGname  => "tsig_name",
    TSIGkey   => "ZX1jVJl7C58GT/sc7Q3uca==",
);

# Configure Nameserver
my $response = $Zone->secondaries([$nameserver,$nameserver1]);
```

Infoblox::DNS::Zone->primary( )

Use this function to specify a nameserver at the zone level.

Example

```
#Construct an object
  my $nameserver = Infoblox::DNS::Nameserver->new(
    name      => "ns4.test.com",
    ipv4addr  => "4.4.5.6",
    stealth   => "false",
    TSIGname  => "tsig_name",
    TSIGkey   => "ZX1jVJl7C58GT/sc7Q3uca==",
);

#Construct an object
  my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name      => "nsv6.test.com",
    ipv6addr  => "2004::223",
    stealth   => "false",
    TSIGname  => "tsig_name",
    TSIGkey   => "ZX1jVJl7C58GT/sc7Q3uca==",
);

# Configure Nameserver
my $response = $Zone->primary([$nameserver,$nameserver1]);
```

Infoblox::Grid::Member->prefer_resolver( )

Use this function to specify a nameserver at the grid member level.

Example

```
#Construct an object
  my $nameserver = Infoblox::DNS::Nameserver->new(
    ipv4addr  => "10.7.5.6",
);

# Configure Nameserver
my $response = $grid_member_object->prefer_resolver([$nameserver]);
```

Infoblox::Grid::Member->alternate_resolver( )

Use this function to specify a nameserver at the grid member level.

Example

```
#Construct an object
  my $nameserver = Infoblox::DNS::Nameserver->new(
    ipv4addr  => "10.10.5.6",
);

# Configure Nameserver
my $response = $grid_member_object->alternate_resolver([$nameserver]);
```

Infoblox::Grid::DNS::Nsgroup->primary( )

Use this function to specify a nameserver at the grid DNS level.

Example

```
#Construct an object
  my $nameserver = Infoblox::DNS::Nameserver->new(
    name      => "ns5.test.com",
    ipv4addr  => "10.10.5.6",
    stealth   => "false",
    TSIGname  => "tsig_name",
    TSIGkey   => "Zx1JVjl7C58gT/sc7Q3uca==",
);

# Configure Nameserver
my $response = $grid_member_object->primary([$nameserver]);
```

Infoblox::Grid::DNS::Nsgroup->secondaries( )

Use this function to specify a nameserver at the grid DNS level.

Example

```
#Construct an object
  my $nameserver = Infoblox::DNS::Nameserver->new(
    name      => "ns5.test.com",
```

```
    ipv4addr  => "10.10.5.6",
    stealth   => "false",
    TSIGname  => "tsig_name",
    TSIGkey   => "Zx1JVjl7C58gT/sc7Q3uca==",
);

# Configure Nameserver
my $response = $grid_member_object->secondaries([$nameserver]);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a Nameserver object.

ddns_zone( )

Use this method to set or retrieve the zone name served by the name server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired ddns_zone for the name server in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting ddns_zone
my $ddns_zone = $nameserver->ddns_zone();
# Modifying ddns_zone
$nameserver->ddns_zone("zone2");
```

gss_tsig_domain( )

Use this method to set or retrieve the domain in which GSS-TSIG for dynamic updates is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired domain name in string format with a maximum of 1023 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting gss_tsig_domain
my $gss_tsig_domain = $nameserver->gss_tsig_domain();
# Modifying gss_tsig_domain
$nameserver->gss_tsig_domain("mydomain");
```

gss_tsig_dns_principal( )

Use this method to set or retrieve the principal name in which GSS-TSIG for dynamic updates is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired principal name in string format with a maximum of 1023 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting gss_tsig_dns_principal
my $gss_tsig_dns_principal = $nameserver->gss_tsig_dns_principal();
# Modifying gss_tsig_dns_principal
$nameserver->gss_tsig_dns_principal("local1/a.localhost@LOCAL");
```

ipv4addr( )

Use this method to set or retrieve the IPv4 address. Required only when ipv6 is not specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting ipv4addr
my $ipv4addr = $nameserver->ipv4addr();
# Modifying ipv4addr
$nameserver->ipv4addr("10.0.0.3");
```

ipv6addr( )

Use this method to set or retrieve the IPv6 address. Required only when ipv4 is not specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv6 address is a 128-bit number in colon hexadecimal notation. It consists of eight groups of four hexadecimal digits separated by colons (example: 12ab:0000:0000:0123:4567:89ab:0000:cdef).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting ipv6addr
my $ipv6addr = $nameserver->ipv6addr();
# Modifying ipv6addr
$nameserver->ipv6addr("12ab::345");
```

name( )

Use this method to set or retrieve a resolvable domain name for the server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name for the name server in FQDN format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting name
my $name = $nameserver->name();
# Modifying name
$nameserver->name("ns2.domain.com");
```

stealth( )

Use this method to make the name server stealth or not. When the attribute stealth is `"true"`, the name server is in stealth mode. This

means that its NS record is not published among the zone data, and it does not respond to queries from resolvers and other name servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "true" or "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting stealth
my $stealth = $nameserver->stealth();
# Modifying stealth
$nameserver->stealth("true");
```

TSIGkey( )

Use this method to set or retrieve the TSIG key. This key must also be present on the external primary server. You can generate a TSIG key, or you can obtain the TSIG key name and key from the external name server either by accessing the appliance yourself or by requesting the appliance administrator to deliver them to you through some out-of-band mechanism. Then type or copy-and-paste that name and key into the appropriate fields.

OR

Use DNS One 2.x TSIG: If you want to use TSIG authentication and the external primary name server is an Infoblox appliance running DNS One 2.x code. The local appliance generates the required TSIG key for authenticating DNS messages to and from appliances running DNS One 2.x code.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired TSIGkey of the name server in string format. Valid values are ":2xCOMPAT" or a TSIG key string.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting TSIGkey
my $TSIGkey = $nameserver->TSIGkey();
# Modifying TSIGname
$nameserver->TSIGkey("ZX1jVJl7C58GT/sc7Q3uc2==");
```

TSIGname( )

Use this method to set or retrieve the name of the TSIG key you want to use.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name of the TSIG key in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting TSIGname
my $TSIGname = $nameserver->TSIGname();
# Modifying TSIGname
$nameserver->TSIGname("tsig2_name");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, modify. This sample also includes error handling for the operations.

##Preparation prior to a DNS nameserver object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a nameserver object

```
 my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name     => "ns1.domain.com",
    ipv4addr => "10.4.5.6",
    stealth  => "false",
    TSIGname => "tsig_name",
    TSIGkey  => "ZX1jVJl7C58GT/sc7Q3ucA==",
);
 unless ($nameserver1){
     die("Construct Nameserver failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}


print "Nameserver object created successfully\n";
```

 # Create a zone and use $nameserver1

```
 #create a zone and use $nameserver1
 my $zone = Infoblox::DNS::Zone->new(
    name    => "domain.com",
    primary => [$nameserver1],
  );

#put zone into session
$session->add($zone)
  or die("Zone creation failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());

print " Zone creation successful\n";
```

#Get and modify the name server

```
#get Zone object from session
my @result = $session->get(
    object => "Infoblox::DNS::Zone",
    name   => "domain.com"
);

unless (scalar(@result) == 0) {
    my $zone = $result[0];

    if ($zone) {

        #modify ipv4addr value
        $nameserver1->ipv4addr("5.5.5.5");
        $zone->primary([$nameserver1]);

        #update session
        $session->modify($zone)
            or die("modify zone domain.com failed: ",
               $session->status_code(), $session->status_detail());
             }
         } else {
```

```
        print "No zone found.";
    }
#####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(),Infoblox::DHCP::View,Infoblox::DNS::Zone, Infoblox::Grid::DHCP, Infoblox::Grid::Member, Infoblox::Grid::DNS::Nsgroup

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Nameserver::Address - NS Name Server object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DNS::Record::NS->nameservers( )
- METHODS
  - address( )
  - auto_create_ptr( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Nameserver::Address - NS Name Server object.

DESCRIPTION

The NS Name Server object provides IP address information for the name server associated with an NS record.

CONSTRUCTOR

```
my $rootns = Infoblox::DNS::Nameserver::Address->new(
             address => $address,                 #Required
             auto_create_ptr => "true" | "false"   #Optional / Default "true"
);
```

MODULE METHODS

You can apply the following functions to an NS Name Server object:

Infoblox::DNS::Record::NS->nameservers( )

Use this function to specify the IP addresses associated with the NS record. See Infoblox::DNS::Record::NS->nameservers() for parameters and return values.

Example

```
#Construct an NS Name Server object
my $ns1 = Infoblox::DNS::Nameserver::Address->new(
             address  => "10.0.0.1"
);

# Configure a name server address on the NS record object
$ns_record->nameservers([$ns1]);
```

METHODS

This section describes all the methods you can use to configure and retrieve the attribute values of an NS Name Server object.

address( )

Use this method to set or retrieve the IPv4 address of the name server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The IPv4 address (32 bits) of the name server.

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you do not specify a parameter, the method returns the attribute value.

Example

```
#Get the address
my $addr = $ns->address();
#Modify the address
$ns->address("10.0.0.1");
```

auto_create_ptr( )

Use this method to set or retrieve the auto_create_ptr flag of the name server. When this flag is set to true, PTR records are automatically created for NS records created in a reverse zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "true" or "false".

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you do not specify a parameter, the method returns the attribute value.

Example

```
#Get auto_create_ptr
my $aptr = $ns->auto_create_ptr();
#Modify auto_create_ptr
$rootns->auto_create_ptr("false");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DNS::Record::NS

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::A - DNS A record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - discovered_name( )
  - discoverer( )
  - first_discovered( )
  - extensible_attributes( )
  - ipv4addr( )
  - last_discovered( )
  - mac_address( )
  - name( )
  - netbios( )
  - network_component_description( )
  - network_component_ip( )
  - network_component_name( )
  - network_component_port_description( )
  - network_component_port_name( )
  - network_component_port_number( )
  - network_component_type( )
  - os( )
  - port_duplex( )
  - port_link_status( )
  - port_speed( )
  - port_status( )
  - port_vlan_description( )
  - port_vlan_name( )
  - port_vlan_number( )
  - ttl( )
  - views( )
  - v_cluster( )
  - v_datacenter( )
  - v_host( )
  - v_name( )
  - v_netadapter( )
  - v_switch( )
  - v_type( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::A - DNS A record object.

DESCRIPTION

An A (address) record maps a domain name to an IPv4 address. To define a specific name-to-address mapping, add an A record to a previously defined authoritative forward-mapping zone.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
my $bind_a = Infoblox::DNS::Record::A->new (
    ipv4addr => $ipv4addr,         #Required
```

```
    name     => $fqdn,             #Required
    comment  => $string,           #Optional / Default is empty
    disable  => "true" | "false"   #Optional / Default "false"
    extensible_attributes => { $string => $string | $num, $string => [ $string | $num, ... ], ... },
#Optional / Default is undefined
    last_discovered  => $string,   #Optional / Default is empty
    mac_address      => $mac_addr, #Optional / Default is empty
    ttl      => $num | undef,      #Optional / Default is undefined
    views    => [$View1],          #Optional / Default is "default" view
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS A object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $bind_a = Infoblox::DNS::Record::A->new (
    ipv4addr => "1.1.1.1",
    name     => "bind_a.domain.com",
    comment  => "this is a demo bind_a record",
    disable  => "true",
    ttl      => "33",
    mac_address      => "1:1:1:1:1:1",
    views    => [$view1]  );
# Submit for addition
my $response = $session->add( $bind_a );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS A object:

  name - Optional. A host name in string format.
  view - Optional. The DNS view in which the A record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  zone - Optional. A zone name in FQDN format.
  ipv4addr - Optional. The address of the host record.
  extensible_attributes - Optional. A hash reference containing extensible attributes.
  v_cluster - Optional. Discovered VMware cluster name.
  v_datacenter - Optional. Discovered VMware datacenter name.
  v_host - Optional. Discovered VMware server name.
  v_name - Optional. Discovered VMware host or virtual machine name.
  v_netadapter - Optional. Discovered VMware physical network adapter name.
  v_switch - Optional. Discovered VMware virtual switch name.
  v_type - Optional. Discovered VMware entity type (possible values: "VirtualMachine", "HostSystem",
"VirtualCenter").
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::A",
    name   => "bind_a.domain.com",
    view   => "default" );

#get all DNS A recods with the extensible attribute 'Site'
my @retrieved_objs = $session->get(
   object => "Infoblox::DNS::Record::A",
   extensible_attributes => { 'Site' => 'Santa Clara' });

# get DNS A records by v_type:
my @retrieved_objs = $session->get(
   object    => 'Infoblox::DNS::Record::A',
   v_type    => 'VirtualMachine');
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$bind_a->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $bind_a );
```

Infoblox::Session->remove( )

Use this method to submit to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::A",
    name   => "bind_a.domain.com",
    view   => "default" );
# Find the desired object from the retrieved list.
my $desired_a = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_a );
```

Infoblox::Session->search( )

Use this method to search for DNS A record objects in the Infoblox appliance. See Infoblox::Session->search() for para meters and return values.

Key References

```
 Apply the following attributes to search for a DNS A record object:

  name     - Optional. A host name in string format (regular expression).
  view     - Optional. The DNS view in which the A record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  zone     - Optional. A zone name in FQDN format.
  comment  - Optional. A comment in string format (regular expression).
  ipv4addr - Optional. The address of the host record (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
  v_cluster - Optional. Discovered VMware cluster name (regular expression).
  v_datacenter - Optional. Discovered VMware datacenter name (regular expression).
  v_host    - Optional. Discovered VMware server name (regular expression).
  v_name    - Optional. Discovered VMware host or virtual machine name (regular expression).
  v_netadapter - Optional. Discovered VMware physical network adapter name (regular expression).
  v_switch  - Optional. Discovered VMware virtual switch name (regular expression).
  v_type    - Optional. Discovered VMware entity type (possible values: "VirtualMachine",
"HostSystem", "VirtualCenter").
```

Example

```
# search for all DNS A objects that match "domain.com" in the default DNS view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::A",
    name   => 'domain\.com',
    view   => "default" );

# search for all DNS A records in the "domain.com" zone of the default view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::A",
    zone   => "domain.com",
    view   => "default" );

#get all DNS A recods with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
  object => "Infoblox::DNS::Record::A",
  extensible_attributes => { 'Site' => 'Santa Clara' });

# search DNS A records by v_datacenter:
my @retrieved_objs = $session->search(
   object       => "Infoblox::DNS::Record::A",
   v_datacenter => "test.*");
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an A record

comment( )

Use this method to set or retrieve the descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $bind_a->comment();
#Modify comment
$bind_a->comment("Modifying the DNS A comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $bind_a->disable();
#Modify disable
$bind_a->disable("true");
```

discovered_name( )

Use this method to retrieve the discovered name of an A Record object.

Parameter

none

Returns

The method returns the discovered name.

Example

```
#Get discovered_name
my $discovered_name = $bind_a->discovered_name();
```

discoverer( )

Use this method to retrieve the discoverer of an A Record object.

Parameter

none

Returns

The method returns the discoverer value.

Example

```
#Get discoverer
my $discoverer = $bind_a->discoverer();
```

first_discovered( )

Use this method to retrieve the date and time that the A Record object was first discovered.

Parameter

none

Returns

The method returns the date and time when the A Record object was first discovered.

Example

```
#Get first_discovered
my $first_discovered = $bind_a->first_discovered();
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $bind_a->extensible_attributes();
#Modify extensible attributes
$bind_a->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

ipv4addr( )

Use this method to set or retrieve the the IPv4 address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192. 168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $bind_a->ipv4addr();
#Modify ipv4addr
$bind_a->ipv4addr("2.2.2.2");
```

last_discovered( )

Use this method to retrieve the time this object was last seen by a discovery job.

The default value for this field is empty.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get last_discovered
my $last_discovered = $bind_a->last_discovered();
```

mac_address( )

Use this method to set or retrieve the the mac address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

MAC Address of the A record object. Format for the hexadecimal strings for MAC addresses and vendor prefixes with colons or dashes. Both of the following formats are acceptable: 11:11:11:11:11:11 and 11-11-11-11-11-11.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mac_address
my $mac_address = $bind_a->mac_address();
#Modify mac_address
$bind_a->mac_address("11::11::11::11::11::12");
```

name( )

Use this method to set or retrieve the host name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Hostname in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the hostname followed by the domain name (example: abc.com). A hostname can have a maximum of 256 bytes.

Infoblox also supports wildcard A records. A wildcard A record maps all the hostnames in a domain to a single IP address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $bind_a->name();
#Modify name
$bind_a->name("bind_a.domain.com");
```

netbios( )

Use this method to retrieve the name in the NetBIOS reply that responded to a NetBIOS query.

Returns

The method returns the attribute value.

Example

```
#Get netbios
my $netbios = $bind_a->netbios();
```

network_component_description( )

Use this method to retrieve the description of the network device that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device description.

Example

```
#Get network_component_description
my $network_component_description = $bind_a->network_component_description();
```

network_component_ip( )

Use this method to retrieve the IP address of the network device that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device IP address.

Example

```
#Get network_component_ip
my $network_component_ip = $bind_a->network_component_ip();
```

network_component_name( )

Use this method to retrieve the name of the network device that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device name.

Example

```
#Get network_component_name
my $network_component_name = $bind_a->network_component_name();
```

network_component_port_description( )

Use this method to retrieve the description of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device port description.

Example

```
#Get network_component_port_description
my $network_component_port_description = $bind_a->network_component_port_description();
```

network_component_port_name( )

Use this method to retrieve the name of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the name of the port on the network device.

Example

```
#Get network_component_port_name
my $network_component_port_name = $bind_a->network_component_port_name();
```

network_component_port_number( )

Use this method to retrieve the number of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device port number in the range 0-9999.

Example

```
#Get network_component_port_number
my $network_component_port_number = $bind_a->network_component_port_number();
```

network_component_type( )

Use this method to retrieve the type of the network device that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device type.

Example

```
#Get network_component_type
my $network_component_type = $bind_a->network_component_type();
```

os( )

Use this method to retrieve the operating system associated with this A record.

Returns

The method returns the attribute value.

Example

```
#Get os
my $os = $bind_a->os();
```

port_duplex( )

Use this method to retrieve the duplex setting of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device port duplex setting. Returned values, if any, are one of the following:

```
Full
Half
```

Example

```
#Get port_duplex
my $port_duplex = $bind_a->port_duplex();
```

port_link_status( )

Use this method to retrieve the link status of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the port link status. Returned values, if any, are one of the following:

```
Connected
Not Connected
Unknown
```

Example

```
#Get port_link_status
my $port_link_status = $bind_a->port_link_status();
```

port_speed( )

Use this method to retrieve the speed of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device port speed value. Returned values, if any, are one of the following:

```
10M
100M
1G
10G
100G
Unknown
```

Example

```
#Get port_speed
my $port_speed = $bind_a->port_speed();
```

port_status( )

Use this method to retrieve the status of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device port status. Returned values, if any, are one of the following:

```
Up
Down
Unknown
```

Example

```
#Get port_status
my $port_status = $bind_a->port_status();
```

port_vlan_description( )

Use this method to retrieve the VLAN description of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the VLAN description of the network device port.

Example

```
#Get port_vlan_description
my $port_vlan_description = $bind_a->port_vlan_description();
```

port_vlan_name( )

Use this method to retrieve the VLAN name of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the VLAN name of the network device port.

Example

```
#Get port_vlan_name
my $port_vlan_name = $bind_a->port_vlan_name();
```

port_vlan_number( )

Use this method to retrieve the VLAN number of the network device port that is connected to the A Record object.

Parameter

none

Returns

The method returns the network device port VLAN number in the range 0-9999.

Example

```
#Get port_vlan_number
my $port_vlan_number = $bind_a->port_vlan_number();
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $bind_a->ttl();
#Modify ttl
$bind_a->ttl(1800);
#Un-override ttl
$bind_a->ttl(undef);
```

views( )

Use this method to set or retrieve the view of the A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" view, which means the A record is located under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $bind_a->views();
#Modify views, list of Infoblox::DNS::View objects
$bind_a->views([$view1]);
```

v_cluster( )

Use this method to retrieve the name of the VMware cluster associated with the A Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware cluster name. The default value is an empty string.

Example

```
#Get v_cluster
my $v_cluster = $bind_a->v_cluster();
```

v_datacenter( )

Use this method to retrieve the name of the VMware datacenter associated with the A Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware datacenter name. The default value is an empty string.

Example

```
#Get v_datacenter
my $v_datacenter = $bind_a->v_datacenter();
```

v_host( )

Use this method to retrieve the name of the VMware host associated with the A Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware host name. The default value is an empty string.

Example

```
#Get v_host
my $v_host = $bind_a->v_host();
```

v_name( )

Use this method to retrieve the name of the VMware entity associated with the A Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware entity name. The default value is an empty string.

Example

```
#Get v_name
my $v_name = $bind_a->v_name();
```

v_netadapter( )

Use this method to retrieve the name of the physical network adapter through which the VMware entity is connected to the appliance. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the network adapter name. The default value is an empty string.

Example

```
#Get v_netadapter
my $v_netadapter = $bind_a->v_netadapter();
```

v_switch( )

Use this method to retrieve the name of the virtual switch through which the VMware entity is connected to the appliance. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the virtual switch name. The default value is an empty string.

Example

```
#Get v_switch
my $v_switch = $bind_a->v_switch();
```

v_type( )

Use this method to retrieve the type of VMware entity associated with the A Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware entity type. The default value is an empty string. The returned value, if any, can be one of the following:

```
VirtualMachine
HostSystem
VirtualCenter
```

Example

```
#Get v_type
my $v_type = $bind_a->v_type();
```

zone( )

Use this method to retrieve the zone name of a DNS A record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $bind_a->zone();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS A record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox Appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";

#Create the zone prior to an A record insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS A record

```
#Construct a DNS A object
my $bind_a = Infoblox::DNS::Record::A->new(
    name     => "bind_a.domain.com",
    comment  => " this is a demo bind_a record ",
    ipv4addr => "1.1.1.1"
);
unless ($bind_a) {
   die("Construct DNS record A failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS A object created successfully\n";

#Add the DNS A record object to Infoblox Appliance through a session
$session->add($bind_a)
    or die("Add record A failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS A object added to server successfully\n";
```

#Search for a specific DNS A record

```
#Search all A records that match "domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::A",
    name   => 'domain\.com'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS A object found at least 1 matching entry\n";

#Search all A records that start with "bind" and end with ".domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::A",
    name   => '^bind.*\.domain\.com\$'
);
my $object = $retrieved_objs[0];
unless ($object) {
```

```
    die("Search record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS A object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS A record

```
#Get A record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::A",
    name   => "bind_a.domain.com"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS A object found at least 1 matching entry\n";

#Modify one of the attributes of the specified A record
$object->ipv4addr("3.3.3.3");

#Apply the changes
$session->modify($object)
    or die("Modify record A failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS A object modified successfully \n";
```

#Remove a DNS A record

```
#Get A record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::A",
    name   => "bind_a.domain.com"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS A object found at least 1 matching entry\n";


#Submit the object for removal
$session->remove($object)
    or die("Remove record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "DNS A object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::AAAA - DNS AAAA record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - ipv6addr( )
  - name( )
  - ttl( )
  - views( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::AAAA - DNS AAAA record object.

DESCRIPTION

An AAAA record, also known as a quad A address record, maps a domain name to an IPv6 address. To define a specific name-to-address mapping, add an AAAA record to a previously defined authoritative forward-mapping zone.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
 my $bindaaaa = Infoblox::DNS::Record::AAAA->new (
     ipv6addr => $ipv6addr,          #Required
     name     => $fqdn,              #Required
     comment  => $string,            #Optional / Default is empty
     extensible_attributes => { $string => $string | $num, $string => [ $string | $num, ... ], ... },
#Optional / Default is undefined
     disable  => "true" | "false"    #Optional / Default "false"
     ttl      => $num | undef,       #Optional / Default is undefined
     views    => [$View1],           #Optional / Default is "default" view
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS AAAA record object.

Infoblox::Session->add( )

Use this method to add a AAAA record object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $bindaaaa = Infoblox::DNS::Record::AAAA->new(
    name     => "aaaa1.domain.com",
    comment  => "add a DNS AAAA aaaa1.domain.com",
    ipv6addr => "aaaa:1111:bbbb:2222:cccc:3333:dddd:4444",
    disable  => "true",
    ttl      => "33",
    views    => [$view1]  );
# Submit for addition
```

```
my $response = $session->add( $bindaaaa );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS AAAA object:

 name - Optional. A host name in string format.
 view - Optional. The DNS view in which the host record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
 zone - Optional. A zone name in FQDN format.
 ipv6addr - Optional. The address of the AAAA record.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
                     object => "Infoblox::DNS::Record::AAAA",
                     name   => "aaaa1.domain.com",
                     view   => "default" );

# get all the DNS AAAA records in the "domain.com" zone of the default view
my @retrieved_objs = $session->get(
                     object => "Infoblox::DNS::Record::AAAA",
                     zone   => "domain.com",
                     view   => "default" );

#get all DNS AAAA recods with the extensible attribute 'Site'
my @retrieved_objs = $session->get(
   object => "Infoblox::DNS::Record::AAAA",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a AAAA record object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$bindaaaa->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $bindaaaa );
```

Infoblox::Session->remove( )

Use this method to remove a AAAA record object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific AAAA record object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::AAAA",
    name   => "aaaa1.domain.com",
    view   => "default" );
# Find the desired object from the retrieved list.
my $desired_aaaa = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_aaaa );
```

Infoblox::Session->search( )

Use this method to search for DNS AAAA record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes, to search for a DNS AAAA record object:

 name     - Optional. A host name in string format (regular expression).
```

```
  view     - Optional. The DNS view in which the AAAA record is located. By default, all DNS views
are searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  zone     - Optional. A zone name in FQDN format.
  comment  - Optional. Comment in string format (regular expression).
  ipv6addr - Optional. The address of the AAAA record (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS AAAA objects that match "domain.com" in the default DNS view
my @retrieved_objs = $session->search(
    object  => "Infoblox::DNS::Record::AAAA",
    name    => 'domain\.com',
    view    => "default",
    comment => ".*modified comment" );

# search for all DNS AAAA records in zone "domain.com" of default view
my @retrieved_objs = $session->search(
                     object => "Infoblox::DNS::Record::AAAA",
                     zone   => "domain.com",
                     view   => "default" );

# search all DNS AAAA recods with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::Record::AAAA",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an AAAA record

comment( )

Use this method to set or retrieve the descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $bindaaaa->comment();
#Modify comment
$bindaaaa->comment("Modifying the DNS AAAA comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $bindaaaa->disable();
#Modify disable
$bindaaaa->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $bindaaaa->extensible_attributes();
#Modify extensible attributes
$bindaaaa->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

ipv6addr( )

Use this method to set or retrieve the the IPv6 address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv6 address is a 128-bit number in colon hexadecimal notation. It consists of eight 16-bit groups of hexadecimal digits separated by colons (example: 12ab:0000:0000:0123:4567:89ab:0000:cdef).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6addr
my ipv6addr = $bindaaaa->ipv6addr();
#Modify ipv6addr
$bindaaaa->ipv6addr("aaaa:1111:bbbb:2222:cccc:3333:dddd:5555");
```

name( )

Use this method to set or retrieve the host name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Hostname in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the hostname followed by the domain name (example: abc.com). A hostname can have a maximum of 256 bytes.

Infoblox also supports wildcard AAAA records. A wildcard AAAA record maps all the hostnames in a domain to a single IP address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $bindaaaa->name();
#Modify name
$bindaaaa-> name("aaaa2.domain.com");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $bindaaaa->ttl();
#Modify ttl
$bindaaaa->ttl(1800);
#Un-override ttl
$bindaaaa->ttl(undef);
```

views( )

Use this method to set or retrieve the view of the AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" view, which means the AAAA record is located under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $bindaaaa->views();
#Modify views, list of Infoblox::DNS::View objects
$bindaaaa->views([$view1]);
```

zone( )

Use this method to retrieve the zone name of a DNS AAAA record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $bindaaaa->zone();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a DNAME record object, such as add, search,

modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS AAAA record insertion

```
#PROGRAM STARTS: Include all the modules that will be used

use strict;
use Infoblox;

#Create a session to the Infoblox Appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2", #appliance host ip
    username => "admin",     #appliance user login
    password => "infoblox"   #appliance password
);

unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

#Create the zone prior to an AAAA record insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");

unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone",
                           name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS AAAA record

```
#Construct a DNS AAAA object

my $bindaaaa = Infoblox::DNS::Record::AAAA->new(
    name     => "aaaa1.domain.com",
    comment  => "add a DNS AAAA aaaa1.domain.com",
    ipv6addr => "aaaa:1111:bbbb:2222:cccc:3333:dddd:4444"
);

unless ($bindaaaa) {
   die("Construct DNS record AAAA failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS AAAA object created successfully\n";

#Add the DNS AAAA record object to Infoblox Appliance through a session
$session->add($bindaaaa)

    or die("Add record AAAA failed: ",
           $session->status_code() . ":" . $session->status_detail());
     print "DNS AAAA object added to server successfully\n";
```

#Search for a specific DNS AAAA record

```
#Search all AAAA records that match "domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::AAAA",
    name   => 'domain\.com'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS AAAA object found at least 1 matching entry\n";
```

```
#Search all AAAA records that start with "add" in the comment
my @retrieved_objs = $session->search(
    object    => "Infoblox::DNS::Record::AAAA",
    comment   => "add.*",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS AAAA object found at least 1 matching entry\n";

#Search all AAAA records that start with "aaaa1" and end with ".com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::AAAA",
    name   => 'aaaa1.*\.com',
 );
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS AAAA object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS AAAA record

```
#Get AAAA record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::AAAA",
    name   => "aaaa1.domain.com"
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

#Modify one of the attributes of the specified AAAA record
$object->ipv6addr("aaaa:1111:bbbb:2222:cccc:3333:dddd:eeee");

#Applying the changes
$session->modify($object)

    or die("Modify record AAAA failed: ",
           $session->status_code() . ":" . $session->status_detail());
     print "DNS AAAA object modified successfully \n";
```

#Remove a DNS AAAA record.

```
#Get AAAA record through the session
my @retrieved_objs = $session->get(
    object   => "Infoblox::DNS::Record::AAAA",
    name     => "aaaa1.domain.com"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Get DNS AAAA object found at least 1 matching entry\n";

#Submit the object for removal

$session->remove($object)
    or die("Remove record AAAA failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS AAAA object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::CNAME - DNS CNAME record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - canonical( )
  - comment( )
  - disable( )
  - extensible_attributes( )
  - name( )
  - ttl( )
  - views( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::CNAME - DNS CNAME record object.

DESCRIPTION

A CNAME record maps an alias to a canonical name. You can use CNAME records in both forward- and IPv4 reverse-mapping zones to serve two different purposes. (At this time, you cannot use CNAME records with IPv6 reverse-mapping zones.)

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
  my $bind_cname =  Infoblox::DNS::Record::CNAME->new (
      canonical => $fqdn,                 #Required
      name      => $fqdn,                 #Required
      comment   => $string,               #Optional / Default is empty
      disable   => "true"|"false"         #Optional / Default "false"
      extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
      ttl       => $num|undef,            #Optional / Default is undefined
      views     => [$View1],              #Optional / Default is "default" view
   );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a CNAME record object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
      #Construct an object
      my $bind_cname = Infoblox::DNS::Record::CNAME->new (
          name      => "bind_cname.domain.com",
          canonical => "cname.domain.com",
          comment   => "this is a demo bind_cname record",
          disable   => "true",
          ttl       => "33",
          views     => [$view1]  );
      # Submit for addition
      my $response = $session->add( $bind_cname );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS CNAME object:

  name - Optional. A host name in string format.
  view - Optional. The DNS view in which the CNAME record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  zone - Optional. A zone name in FQDN format.
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::CNAME",
    name   => "bind_cname.domain.com",
    view   => "default" );

# get all DNS CNAME record in zone "domain.com" of default view
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::CNAME",
    zone   => "domain.com",
    view   => "default" );

# get all DNS CNAME record with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::CNAME",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$bind_cname->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $bind_cname );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Get the objects with the same name
my @retrieved_objs = $session->get(
  object => "Infoblox::DNS::Record::CNAME",
  name   => "bind_cname.domain.com",
  view   => "default");
#Find the desired object from the retrieved list.
my $desired_cname = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_cname );
```

Infoblox::Session->search( )

Use this method to search for DNS CNAME record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific DNS CNAME object.

  name    - Optional. A host name in string format (regular expression).
  view    - Optional. The DNS view in which the CNAME record is located. By default, all DNS views
are searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  zone    - Optional. A zone name in FQDN format.
  comment - Optional. A comment in string format (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS CNAME objects that match "domain.com" in the default DNS view
my @retrieved_objs = $session->search(
    object  => "Infoblox::DNS::Record::CNAME",
    name    => 'domain\.com',
    view    => "default",
    comment => "this is a modified comment" );

# search for all DNS CNAME record in zone "domain.com" of default view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::CNAME",
    zone   => "domain.com",
    view   => "default" );

# search all DNS CNAME recods with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::Record::CNAME",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a CNAME record.

canonical( )

Use this method to set or retrieve the canonical (or official) name of the host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Canonical name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the hostname followed by the domain name (example: abc.com). A hostname can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get canonical
my $canonical = $bind_cname->canonical();
#Modify canonical name
$bind_cname->canonical("bind_cname.domain.com");
```

comment( )

Use this method to set or retrieve the descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $bind_cname->comment();
#Modify comment
$bind_cname->comment("Modifying the DNS CNAME comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $bind_cname->disable();
#Modify disable
$bind_cname->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS CNAME record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $bind_cname->extensible_attributes();
#Modify extensible attributes
$bind_cname->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

name( )

Use this method to set or retrieve the host name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host_name followed by the domain name (example: abc.com). A host_name can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $bind_cname->name();
#Modify name
$bind_cname->name("bind_cname2.domain.com");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $bind_cname->ttl();
#Modify ttl
$bind_cname->ttl(1800);
#Un-override ttl
$bind_cname->ttl(undef);
```

views( )

Use this method to set or retrieve the view of the CNAME record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" view, which means the CNAME record is located under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $bind_cname->views();
#Modify views, list of Infoblox::DNS::View objects
$bind_cname->views([$view1]);
```

zone( )

Use this method to retrieve the zone name of a DNS CNAME record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $bind_cname->zone();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample code also includes error handling for the operations.

#Preparation prior to a DNS CNAME record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
```

```
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the zone prior to a CNAME record insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS CNAME record

```
my $view1 = Infoblox::DNS::View->new(
    name      => "default" );

#Construct a DNS CNAME object
my $bind_cname = Infoblox::DNS::Record::CNAME->new(
    name      => "bind_cname.domain.com",
    canonical => "cname.domain.com",
    comment   => "this is a demo bind_cname record",
    ttl       => "33",
    views     => [$view1]);
unless ($bind_cname) {
   die("Construct DNS record CNAME failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS CNAME object created successfully\n";

#Add the DNS CNAME record object to the Infoblox appliance through a session
$session->add($bind_cname)
    or die("Add record cname failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS CNAME object added to server successfully\n";
```

#Search for a specific DNS CNAME record

```
#Search all CNAME records that match "domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::CNAME",
    name   => 'domain\.com'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record CNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS CNAME object found at least 1 matching entry\n";

#Search all CNAME records that start with "bind" and end with ".domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::CNAME",
    name   => '^bind.*\.domain\.com\$'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record CNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
```

```
}
print "Search DNS CNAME object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS CNAME record

```
#Get CNAME record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::CNAME",
    name   => "bind_cname.domain.com",
    view   => "default"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record CNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS CNAME object found at least 1 matching entry\n";

#Modify one of the attributes of the specified CNAME record
$object->disable("true");

#Apply the changes
$session->modify($object)
    or die("Modify record CNAME failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS CNAME object modified successfully \n";
```

#Remove a DNS CNAME record

```
#Get CNAME record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::CNAME",
    name   => "bind_cname.domain.com",
    view   => "default"
);
$object = $retrieved_objs[0];
unless ($object) {
    die("Get record CNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS CNAME object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove( $object )
  or die("Remove record CNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "DNS CNAME object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::DNAME - DNAME record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - name( )
  - target( )
  - ttl( )
  - views( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::DNAME - DNAME record object.

DESCRIPTION

A DNAME record maps all the names in one domain to those in another domain, essentially substituting one domain name suffix with the other.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
 my $bind_dname = Infoblox::DNS::Record::DNAME->new (
     name    => $fqdn,                                                          #Required
     target  => $fqdn,                                                          #Required
     comment => $string,                                                        #Optional /
Defauilt is empty
     disable => "true" | "false",                                               #Optional /
Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl     => $num | undef,                                                   #Optional /
Default is undefined
     views   => [$View1],                                                       #Optional /
Default is "default" view
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS DNAME object.

Infoblox::Session->add( )

Use this method to add a DNAME record object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct object
my $bind_dname = Infoblox::DNS::Record::DNAME->new (
    name    => "bind_dname.domain.com",
    comment => "add dname bind_dname.domain.com",
    target  => "target.org",
```

```
        disable => "true",
        ttl     => "33",
        views   => [$view1]
        );

    # Submit for addition
    my $response = $session->add( $bind_dname );
```

Infoblox::Session->get( )

Use this method to retrieve existing DNAME record objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
    Apply the following attributes to get a specific DNS DNAME object:

     name - Optional. A domain name in string format (regular expression).
     view - Optional. The DNS view in which the DNAME record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
     zone - Optional. A zone name in FQDN format.
     extensible_attributes - Optional. A hash reference containing extensible attributes.

=item B<Example>

    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::DNAME",
        name   => "bind_dname.domain.com",
        view   => "default"
        );

    # get all DNS DNAME record in zone "domain.com" of default view
    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::DNAME",
        name   => "domain.com",
        view   => "default"
        );

    # get all DNS DNAME record with a given extensible attribute
    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::DNAME",
        extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify DNAME record objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
    # Use method to modify the comment.
    $bind_dname->comment("this is a modified comment.");
    # Submit modification
    my $response = $session->modify( $bind_dname );
```

Infoblox::Session->remove( )

Use this method to remove a DNAME record object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
    # Get the objects with the same name
    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::DNAME",
        name   => "bind_dname.domain.com",
        view   => "default" );
    # Find the desired object from the retrieved list.
    my $desired_dname = $retrieved_objs[0];
    # Submit for removal
    my $response = $session->remove( $desired_dname );
```

Infoblox::Session->search( )

Use this method to search for existing DNAME record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters

and return values.

Key Reference

```
Apply the following attributes, to search for a DNS DNAME record object:

  name    - Optional. A domain name in string format (regular expression).
  view    - Optional. The DNS view in which the DNAME record is located. By default, all DNS views
are searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  comment - Optional. A comment in string format (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS DNAME objects that match "domain.com" in the default DNS view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::DNAME",
    name   => '.*\.domain\.com',
    view   => "default"
    );

# search for all DNS DNAME records in the "domain.com" zone of the default view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::DNAME",
    zone   => "domain.com",
    view   => "default"
    );

# search all DNS DNAME recods with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::Record::DNAME",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DNAME record.

comment( )

Use this method to set or retrieve the comment value.

Enter identifying text for this record, such as a meaningful note or reminder.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $binddname->comment();
#Modify comment
$binddname->comment("Modifying the DNAME comment");
```

disable( )

Use this method to set or retrieve the disable value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is "false", therefore the DNS record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $binddname->disable();
#Modify disable
$binddname->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS DNAME record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $binddname->extensible_attributes();
#Modify extensible attributes
$binddname->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

name( )

Use this method to set or retrieve the name value.

Enter the name of a sub domain. If you are adding a DNAME record for the entire zone, leave this field empty. This field is for adding a DNAME record for a sub domain within the selected zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Hostname in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the hostname followed by the domain name (example: abc.com). A hostname can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $binddname->name();
#Modify name
$binddname->name("bind_dname1.domain.com");
```

target( )

Use this method to set or retrieve the target value.

Enter the domain name to which you want to map all the domain names specified in the Domain Name field.

Include the specified parameter to set the attribute value. Must use some parameter to add a DNAME record.

Parameter

Text with the target (domain-name) for the DNS DNAME record.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $target = $binddname->target();
#Modify name
$binddname->target("modified_target.org");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined; therefore the record inherits the TTL value from the zone level.

With a specified TTL value, the method overrides the zone values with the specified TTL value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $binddname->ttl();
#Modify ttl
$binddname->ttl(1800);
#Un-override ttl
$binddname->ttl(undef);
```

views( )

Use this method to set or retrieve the views value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the default view, therefore the DNAME record is referenced under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $binddname->views();
#Modify views, list of Infoblox::DNS::View objects
$binddname->views([$view1]);
```

zone( )

Use this method to retrieve the zone name of a DNS DNAME record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $binddname->zone();
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to a DNAME record object, such as create, search, modify, and remove an object. This sample code also includes error handling for the operations.

#Preparation prior to a DNAME record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the zone prior to a DNAME record insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNAME record

```
#Construct a DNAME object
my $bind_dname = Infoblox::DNS::Record::DNAME->new(
    name     => "bind_dname.domain.com",
    target   => "target.org",
    comment  => "this is a demo bind_dname record",
);
unless ($bind_dname) {
   die("Construct DNS record DNAME failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS DNAME object created successfully\n";

#Add the DNS DNAME record object to Infoblox appliance through a session
$session->add($bind_dname)
    or die("Add record DNAME failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS DNAME object added to server successfully\n";
```

#Search for a specific DNS DNAME record

```
#Search all DNAME records in the zone
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::DNAME",
    name   => 'domain\.com'
);
my $object = $retrieved_objs[0];
unless  $object
```

```
        (        ) {
    die("Search record DNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS DNAME object found at least 1 matching entry\n";

#Search all DNAME records that start with "bind" and end with "domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::DNAME",
    name   => '^bind.*\.domain.com\$'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record DNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS DNAME object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS DNAME record

```
#Get DNAME record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::DNAME",
    name   => "bind_dname.domain.com",
    target => "target.org"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record DNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS DNAME object found at least 1 matching entry\n";

#Modify one of the attributes of the specified DNAME record
$object->comment("This is a modified comment.");

#Apply the changes
$session->modify($object)
    or die("Modify record DNAME failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS DNAME object modified successfully \n";
```

#Remove a DNS DNAME record

```
#Get DNAME record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::DNAME",
    name   => "bind_dname.domain.com"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record DNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS DNAME object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove record DNAME failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "DNS DNAME object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::DNSKEY - DNSKEY Record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
- METHODS
  - algorithm( )
  - flags( )
  - key_tag( )
  - name( )
  - public_key( )
  - ttl( )
  - view( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::DNSKEY - DNSKEY Record object.

DESCRIPTION

The DNSKEY resource record stores public keys for the DNSSEC authentication process. The DNSKEY records are generated automatically when the corresponding authoritative zone is signed. The the DNSKEY resource record object is read-only.

The DNSKEY resource record is defined in RFC 4034.

CONSTRUCTOR

The Infoblox::DNS::Record::DNSKEY object is a read-only object and does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to a DNSKEY Record object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get the DNSKEY Record object(s):

     algorithm - Optional. The public key encryption algorithm. The allowed values are
'1','3','5','6','7','RSAMD5','RSASHA1','DSA','NSEC3RSASHA1','NSEC3DSA','RSASHA256','RSASHA512','NSEC3RS
and'NSEC3RSASHA512'.
     flags     - Optional. Flags field to identify the type of the key.
     key_tag   - Optional. Key tag to identify the public key.
     name      - Optional. FQDN of the DNSKEY record.
     public_key- Optional. Public key in the base-64 encoding.
     ttl       - Optional. TTL of the DNSKEY record.
     view      - Optional. The DNS view in which the record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
     zone      - Optional. A zone name in FQDN format.
```

Example

```
# get all the DNSKEY records for the zone 'domain.com' in the default view.
my @retrieved_objs = $session->get(
                     object => "Infoblox::DNS::Record::DNSKEY",
                     name   => "domain.com",
                     view   => "default" );

# get all the DNSKEY records in the "domain.com" zone of the default view
```

```
# with algorithm field having the value "RSASHA1" and "flags" having value 257
my @retrieved_objs = $session->get(
                     object    => "Infoblox::DNS::Record::DNSKEY",
                     name      => "domain.com",
                     algorithm => "RSASHA1",
                     flags     => 257,
                     view      => "default" );
```

Infoblox::Session->search( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to search the DNSKEY Record object(s):

     algorithm - Optional. The public key encryption algorithm. The allowed values are
'1','3','5','6','7','RSAMD5','RSASHA1','DSA','NSEC3RSASHA1','NSEC3DSA','RSASHA256','RSASHA512','NSEC3RS
and 'NSEC3RSASHA512'.
     flags     - Optional. Flags field to identify the type of the key.
     key_tag   - Optional. Key tag to identify the public key.
     name      - Optional. FQDN of the DNSKEY record (regular expression).
     public_key- Optional. Public key in the base-64 encoding (regular expression).
     ttl       - Optional. TTL of the DNSKEY record.
     view      - Optional. The DNS view in which the record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
     zone      - Optional. A zone name in FQDN format.
```

Example

```
# search the DNSKEY records in the default view by zone name.
my @retrieved_objs = $session->search(
                     object => "Infoblox::DNS::Record::DNSKEY",
                     name   => ".*[.]com",
                     view   => "default" );

# search for all DNSKEY records in the "domain.com" zone of the default view
# Note that the 'default' view is assumed implicitly here
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::DNSKEY",
    zone   => "domain.com",
   );
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of a DNSKEY Record object.

algorithm( )

Use this method to retrieve the public key encryption algorithm of a DNSKEY Record object.

Parameter

none

Returns

The method returns the public key encryption algorithm attribute value. Returned values are one of the following:

RSAMD5
DSA
RSASHA1
NSEC3DSA
NSEC3RSASHA1
RSASHA256
RSASHA512

Example

```
#Get the public key encryption algorithm.
my $algorithm = $dnskey->algorithm();
```

flags( )

Use this method to retrieve the flags of a DNSKEY Record object.

Parameter

none

Returns

The method returns the flags attribute value. The flags field is a 16-bit unsigned integer. Currently, only two bits of this value are used--the least significant bit and bit 7. The other bits are reserved for future use and must be zero. If bit 7 is set to 1, the key is a DNS zone key. Otherwise, the key is not a zone key and cannot be used to verify zone data. The least significant bit indicates "secure entry point property". If it is not zero, the key is a key signing key (KSK type). Otherwise, the key type is ZSK.

Example

```
#Getting flags
my $flags = $dnskey->flags();
```

key_tag( )

Use this method to retrieve the key tag identifying the public key of a DNSKEY Record object.

Parameter

none

Returns

The method returns the key tag of the public key value. A key tag is a 16-bit integer key fingerprint. It is used to distinguish one key from another when multiple keys are associated with the same zone.

Example

```
#Get the key tag of the public key
my $key_tag = $dnskey->key_tag();
```

name( )

Use this method to retrieve the FQDN of the zone that owns the DNSKEY record.

Parameter

none

Returns

The method returns the FQDN of the zone that contains the given DNSKEY record.

Example

```
#Get the FQDN of the DNSKEY record
my $name = $dnskey->name();
```

public_key( )

Use this method to retrieve the public key. The format of the returned value depend on the key algorithm.

Parameter

none

Returns

The method returns the public key. The exact format of the returned value depends on the algorithm of the key.

Example

```
#Get public key.
my $public_key = $dnskey->public_key();
```

ttl( )

Use this method to retrieve the Time to Live (TTL) value of a DNSKEY Record object.

Parameter

none

Returns

The method returns the TTL attribute value. The returned parameter is a 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the record is cached. Zero indicates that the record should not be cached.

Example

```
#Get the TTL value
my $ttl = $dnskey->ttl();
```

view( )

Use this method to retrieve the DNS View object that contains the DNSKEY Record object.

Parameter

none

Returns

The method returns the Infoblox::DNS::View object that contains the DNSKEY record.

Example

```
#Get the view
my $view = $dnskey->view();
```

zone( )

Use this method to retrieve the zone name of a DNSKEY record.

Parameter

none

Returns

The method returns the name of the zone that contains the given DNSKEY record. The returned value is a string in FQDN format.

Example

```
# Get zone
my $zone = $dnskey->zone();
```

SAMPLE CODE

The following sample code demonstrates the session methods on a DNSKEY Record object.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device
       my $session = Infoblox::Session->new(
           master   => "192.168.1.2",
           username => "admin",
           password => "infoblox"
       );
       unless ($session) {
          die("Construct session failed: ",
              $session->status_code() . ":" . $session->status_detail());
       }
       print "Session created successfully\n";

       #Enable DNSSEC in the default view

       my $default_view=$session->get(
                                   object=> "Infoblox::DNS::View",
                                   name => "default"
                                  );
       unless($default_view) {
           die("Getting the default view failed:",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
       print "Got the default view successfully\n";

       $default_view->dnssec_enabled("true")
           or die("Changing the dnssec_enabled in the default view failed:",
                  Infoblox::status code            Infoblox::status detail
```

```
                                    () . ":" .                        ());
$session->modify($default_view)
    or die("Changing the dnssec_enabled in the default view failed:",
            Infoblox::status_code() . ":" . Infoblox::status_detail());

#Create the zone

print "Creating Member primary server for the zone\n";
my $primary=Infoblox::DNS::Member->new(
                                        ipv4addr => "192.168.1.2",
                                        name => "infoblox.localdomain"
                                      );
unless($primary) {
   die("Unable to create primary server object: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

my $zone = Infoblox::DNS::Zone->new(
                                      name => "domain.com",
                                      primary => $primary
                                    );
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";

#Retrieving zone back from the server in order to sign it
$zone = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless($zone) {
   die("Retrieving zone back failed: ",
        Infoblox::status_code( ). ":". Infoblox::status_detail( ));
}
print "Zone retrieved for signing successfully.\n";

$zone->dnssec_ksk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_zsk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_ksk_size(640) &&
$zone->dnssec_zsk_size(640)
  or die("Changing the zone DNSSEC setting failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->modify($zone)
  or die("Modifying dnssec values in zone failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Zone modified successfully\n";

#Signing the zone
$zone->dnssec_signed("true")
   or die("Signing of the zone failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
       print "Zone signed successfully\n";
```

#Getting the DNSKEY record

```
my $ksk_dnskey=$session->get(
                             object => "Infoblox::DNS::Record::DNSKEY",
                             flags  => 257,
                             name   => "domain.com",
                             view   => "default"
                            );
     unless($ksk_dnskey) {
 die("Getting ksk_dnskey failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
     }
     print "Got Key-signing key DNSKEY record successfully\n";
     print "Key value: ".$ksk_dnskey->public_key()."\n";
```

```
my $zsk_dnskey=$session->get(
                              object => "Infoblox::DNS::Record::DNSKEY",
                              flags  => 256,
                              name   => "domain.com",
                              view   => "default"
                             );
unless($zsk_dnskey) {
   die("Getting zsk_dnskey failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Got Zone-signing key DNSKEY record successfully\n";
print "Key value: ".$zsk_dnskey->public_key()."\n";
```

#Searching for DNSKEY object

```
my @retrieved_objs=$session->search(
                                    object => "Infoblox::DNS::Record::DNSKEY",
                                    name   => ".*com",
                                    view   => "default"
                                  );
     unless(@retrieved_objs>0) {
   die("Searching for DNSKEY objects failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
     }

#Removing the created zone and cleaning up the view
$session->remove($zone)
  or die("Unable to remove the zone: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Zone removed successfully\n";

$default_view->dnssec_enabled("false")
&& $default_view->override_dnssec("false")
&& $session->modify($default_view)
 or die("Restoring dnssec_enabled value in the default view failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::DNS::Record::DS, Infoblox::DNS::Record::RRSIG, Infoblox::DNS::View, Infoblox::DNS::Zone

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::DS - DS Record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - algorithm( )
  - digest( )
  - digest_type( )
  - key_tag( )
  - name( )
  - ttl( )
  - view( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::DS - DS Record object.

DESCRIPTION

The DS key record is a part of the DNS security extension records. It is used to authorize the DNSKEY records of the child zone and thus to establish the DNSSEC chain of trust.

The DS resource record is defined in RFC 4034.

The DS resource records are automatically generated upon the signing of the child zone of an authoritative zone residing on the appliance. You can also upload the external DS records via Infoblox::Session->import_data(). It is not possible to add, remove or modify automatically generated DS records. However, the records imported with Infoblox::Session->import_data() method can be deleted.

CONSTRUCTOR

The Infoblox::DNS::Record::DS object is a read-only object and does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to an DS Record object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
 Apply the following attributes to get the DS Record object(s):

     algorithm       - Optional. The encryption algorithm of the corresponding DNSKEY record.
                       The allowed values are
'1','3','5','6','7','RSAMD5','RSASHA1','DSA','NSEC3RSASHA1','NSEC3DSA','RSASHA256','RSASHA512','NSEC3RS
and 'NSEC3RSASHA512'.
     digest          - Optional. The digest of the DNSKEY RR.
     digest_type     - Optional. Identifies the algorithm used to construct the digest. Allowed
values are 'SHA1', 'SHA256', 1, 2.
     key_tag         - Optional. Key-tag to identify the public key.
     name            - Optional. FQDN of the DS record.
     ttl             - Optional. TTL value of the record.
     view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
     zone            - Optional. A zone name in FQDN format.
```

Example

```
# get all the DS records for the zone 'domain.com' in the default DNS view
my @retrieved_objs = $session->get(
                     object => "Infoblox::DNS::Record::DS",
                     name   => "domain.com",
                     view   => "default" );

# get all the DS records in the "domain.com" zone of the default DNS view
# with digest type "SHA1" and TTL of 1 day:
my @retrieved_objs = $session->get(
                     object    => "Infoblox::DNS::Record::DS",
                     name      => "domain.com",
                     digest_type => "SHA1",
                     ttl       =>  86400,
                     view      => "default" );
```

Infoblox::Session->remove( )

Use this method to submit an object for removal from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the object, and then submit it for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::DS",
    name   => "domain.com",
    view   => "default" );
# Find the desired object from the retrieved list.
my $desired_ds = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_ds );
```

Infoblox::Session->search( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to search the DS Record object(s):
    algorithm       - Optional. The encryption algorithm of the corresponding DNSKEY record.
                      The allowed values are
'1','3','5','6','7','RSAMD5','RSASHA1','DSA','NSEC3RSASHA1','NSEC3DSA','RSASHA256','RSASHA512','NSEC3RS
and 'NSEC3RSASHA512'.
    digest          - Optional. Digest of the DNSKEY RR (regular expression).
    digest_type     - Optional. Identifies the algorithm used to construct the digest. Allowed
values are 'SHA1', 'SHA256', 1, 2.
    key_tag         - Optional. Key-tag to identify the public key.
    name            - Optional. FQDN of the DS record (regular expression).
    ttl             - Optional. TTL value of the record.
    view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
    zone            - Optional. A zone name in FQDN format.
```

Example

```
# search the DS records in the default DNS view by zone name.
my @retrieved_objs = $session->search(
                     object => "Infoblox::DNS::Record::DS",
                     name   => ".*[.]com",
                     view   => "default" );

# search for all DS records in the "domain.com" zone of the default DNS view
# Note that the 'default' view is assumed implicitly here
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::DS",
    zone   => "domain.com",
   );
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of a DS Record object.

algorithm( )

Use this method to retrieve the public key encryption algorithm of the DNSKEY record of the given DS record.

Parameter

none

Returns

The method returns the public key encryption algorithm attribute value. Returned values are one of the following:

RSAMD5
DSA
RSASHA1
NSEC3DSA
NSEC3RSASHA1
RSASHA256
RSASHA512

Example

```
#Get the public key encryption algorithm.
my $algorithm = $ds_record->algorithm();
```

digest( )

Use this method to retrieve the digest of the DNSKEY resource record that is stored in a DS Record object.

Parameter

none

Returns

The method returns the digest of the corresponding DNSKEY RR attribute value.

Example

```
#Get the digest of the DNSKEY RR
my $digest = $ds_record->digest();
```

digest_type( )

Use this method to retrieve the algorithm that is used to construct the digest stored in the digest field.

Parameter

none

Returns

The method returns the algorithm that is used to construct the digest attribute value.

Example

```
#Get the algorithm used to construct the digest.
my $digest_type = $ds_record->digest_type();
```

key_tag( )

Use this method to retrieve the key tag of the DNSKEY RR that is referred by the given DS Record object.

Parameter

none

Returns

The method returns the key tag attribute value. The key tag is used to identify the public keys that belong to the different DS records. The returned value is an unsigned 16 bit integer.

Example

```
#Get key-tag
my $key_tag = $ds_record->key_tag();
```

name( )

Use this method to retrieve the FQDN of the DNSKEY record referred by the DS Record object.

Parameter

none

Returns

The method returns the FQDN of the DNSKEY record that corresponds to the DS record.

Example

```
#Get  FQDN of the DS record
my $name = $ds_record->name();
```

ttl( )

Use this method to retrieve the Time to Live (TTL) value of a DS Record object.

Parameter

none

Returns

The method returns the TTL attribute value. The returned parameter is a 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the record is cached. Zero indicates that the record should not be cached.

Example

```
#Get  TTL
my $ttl = $ds_record->ttl();
```

view( )

Use this method to retrieve the DNS view object that contains the DS Record object.

Parameter

none

Returns

The method returns the Infoblox::DNS::View object that contains the DS record.

Example

```
#Get  Ref to the view object
my $view = $ds_record->view();
```

zone( )

Use this method to retrieve the zone name of a DS record.

Parameter

none

Returns

Returns the zone name in FQDN format.

Example

```
# Get zone
my $zone = $ds_record->zone();
```

SAMPLE CODE

The following sample code demonstrates the session methods on a DS Record object.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device
       my $session = Infoblox::Session->new(
           master   => "192.168.1.2",
           username => "admin",
```

```
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";

#Enable DNSSEC in the default view

my $default_view=$session->get(
                               object=> "Infoblox::DNS::View",
                               name => "default"
                              );
unless($default_view) {
    die("Getting the default view failed:",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Got the default view successfully\n";

$default_view->dnssec_enabled("true")
    or die("Changing the dnssec_enabled in the default view failed:",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->modify($default_view)
    or die("Changing the dnssec_enabled in the default view failed:",
            Infoblox::status_code() . ":" . Infoblox::status_detail());

#Create the zone

print "Creating Member primary server for the zone\n";
my $primary=Infoblox::DNS::Member->new(
                                       ipv4addr => "192.168.1.2",
                                       name => "infoblox.localdomain"
                                      );
unless($primary) {
   die("Unable to create primary server object: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

my $zone = Infoblox::DNS::Zone->new(
                                    name => "domain.com",
                                    primary => $primary
                                   );
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";

#Retrieving zone back from the server in order to sign it
$zone = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless($zone) {
   die("Retrieving zone back failed: ",
        Infoblox::status_code( ). ":". Infoblox::status_detail( ));
}
print "Zone retrieved for signing successfully.\n";

$zone->dnssec_ksk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_zsk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_ksk_size(640) &&
$zone->dnssec_zsk_size(640)
  or die("Changing the zone DNSSEC setting failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->modify($zone)
  or die("Modifying dnssec values in zone failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
        print "Zone modified successfully\n";

        #Signing the zone
        $zone->dnssec_signed("true")
           or die("Signing of the zone failed: ",
                   Infoblox::status_code() . ":" . Infoblox::status_detail());
               print "Zone signed successfully\n";

       # In order to have DS records in the parent zone we need to have a signed subzone
       my $subzone = Infoblox::DNS::Zone->new(
                                          name => "subzone.domain.com",
                                          primary => $primary
                                        );
        unless ($subzone) {
           die("Construct subzone failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
        }
        print "Zone object created successfully\n";

        #Verify if the subzone exists
        my $object = $session->get(object => "Infoblox::DNS::Zone", name => "subzone.domain.com");
        unless ($object) {
           print "Subzone does not exist on server, safe to add the subzone\n";
           $session->add($subzone)
              or die("Add subzone failed: ",
                     $session->status_code() . ":" . $session->status_detail());
        }
        print "Subzone added successfully\n";

        #Retrieving subzone back from the server in order to sign it
        $subzone = $session->get(object => "Infoblox::DNS::Zone", name => "subzone.domain.com");
        unless($subzone) {
           die("Retrieving subzone back failed: ",
                Infoblox::status_code( ). ":". Infoblox::status_detail( ));
        }
        print "Subzone retrieved for signing successfully.\n";

        $subzone->dnssec_ksk_algorithm("NSEC3DSA") &&
        $subzone->dnssec_zsk_algorithm("NSEC3DSA") &&
        $subzone->dnssec_ksk_size(640) &&
        $subzone->dnssec_zsk_size(640)
          or die("Changing the subzone DNSSEC setting failed: ",
                 Infoblox::status_code() . ":" . Infoblox::status_detail());
        $session->modify($subzone)
          or die("Modifying dnssec values in subzone failed: ",
                 Infoblox::status_code() . ":" . Infoblox::status_detail());
        print "Subzone modified successfully\n";

        #Signing the subzone
        $subzone->dnssec_signed("true")
           or die("Signing of the subzone failed: ",
                   Infoblox::status_code() . ":" . Infoblox::status_detail());
               print "Subzone signed successfully\n";

       #Getting the DNSKEY record for subzone
       my $ksk_dnskey=$session->get(
                                  object => "Infoblox::DNS::Record::DNSKEY",
                                  flags  => 257,
                                  name   => "subzone.domain.com",
                                  view   => "default"
                                 );
            unless($ksk_dnskey) {
         die("Getting ksk_dnskey failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
            }
            print "Got Key-signing key DNSKEY record successfully\n";

#Retrieving the DS record corresponding to the given DNSKEY

      my $ds_record=$session->get(
                                  object => "Infoblox::DNS::Record::DS",
                                  name   => "subzone.domain.com",
                                  key_tag => $ksk_dnskey->key_tag(),
                                  view   => "default"
                                 );
      unless($ds_record) {
```

```
        die("Getting DS record corresponding to ksk_dnskey failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
      }
      print "Got the DS record corresponding to public key with key_tag ".$ksk_dnskey->key_tag()."
successfully\n";
      print "Digest value: ".$ds_record->digest()."\n";

#Searching for DS object

      my @retrieved_objs=$session->search(
                                          object => "Infoblox::DNS::Record::DS",
                                          name   => ".*com",
                                          view   => "default"
                                        );
          unless(@retrieved_objs>0) {
        die("Searching for DS objects failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
          }
          print scalar(@retrieved_objs)." DS objects successfully found by search\n";

#Removing the created zones and cleaning up the view

      $session->remove($subzone)
        or die("Unable to remove the subzone: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
      print "Subzone removed successfully\n";

      $session->remove($zone)
        or die("Unable to remove the zone: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
      print "Zone removed successfully\n";

      $default_view->dnssec_enabled("false")
      && $default_view->override_dnssec("false")
      && $session->modify($default_view)
       or die("Restoring dnssec_enabled value in the default view failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->remove(), Infoblox::DNS::Record::DNSKEY, Infoblox::DNS::View, Infoblox::DNS::Zone

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::MX - DNS MX record object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - exchanger( )
  - extensible_attributes( )
  - name( )
  - pref( )
  - ttl( )
  - views( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::MX - DNS MX record object

DESCRIPTION

An MX (mail exchanger) record maps a domain name to a mail exchanger. A mail exchanger is a server that either delivers or forwards mail. You can specify one or more mail exchangers for a zone, as well as the priority for using each mail exchanger. A standard MX record applies to a particular domain or subdomain. A wildcard MX record applies to a domain and all its subdomains.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
 my $mx = Infoblox::DNS::Record::MX->new(
     exchanger => $fqdn,                                                    #Required
     name      => $fqdn,                                                    #Required
     pref      => $num,                                                     #Required
     comment   => $string,                                                  #Optional /
Default is empty
     disable   => "true" | "false",                                         #Optional /
Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl       => $num | undef,                                             #Optional /
Default is undefined
     views     => [$View1]                                                  #Optional /
Default is "default" view
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS MX object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $bindmx = Infoblox::DNS::Record::MX->new(
    name      => "bind_mx.domain.com",
```

```
        comment   => "add mx bind_mx.domain.com",
        pref      => 1,
        exchanger => "exchanger.domain.com",
    );
    # Submit for addition
    my $response = $session->add( $bindmx );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
    Apply the following attributes to get a specific DNS MX object:

     name      - Optional.  A domain name in string format
     view      - Optional.  The DNS view in which the MX record is located. By default, all DNS views
   are searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
   view only.
     zone      - Optional. A zone name in FQDN format.
     exchanger - Optional. The exchanger of the MX record.
     pref      - Optional. The Preference of the MX record.
     extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
     my @retrieved_objs = $session->get(
        object    => "Infoblox::DNS::Record::MX",
        name      => "bind_mx.domain.com",
        view      => "default" );

     # get all DNS MX records in the "domain.com" zone of the default view
     my @retrieved_objs = $session->get(
        object    => "Infoblox::DNS::Record::MX",
        zone      => "domain.com",
        view      => "default" );

    # get all DNS MX records with a given extensible attribute
    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::MX",
        extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
    #Use method to modify the comment.
    $bindmx->comment("this is a modified comment");
    # Submit modification
    my $response = $session->modify( $bindmx );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
    # Get the object under the same name
    my @retrieved_objs = $session->get(
        name      => "bind_mx.domain.com",
        comment   => "add mx bind_mx.domain.com",
        pref      => 1,
        exchanger => "exchanger.domain.com",
        view      => "default" );
    # find the desired object from retrieved list.
    my $desired_mx = $retrieved_objs[0];
    # Submit for removal
    my $response = $session->remove( $desired _mx );
```

Infoblox::Session->search( )

Use this method to search for DNS MX record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS MX record object:

  name      - Optional. A domain name in string format (regular expression).
  view      - Optional. The DNS view in which the MX record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  zone      - Optional. A zone name in FQDN format.
  comment   - Optional. A comment in string format (regular expression).
  exchanger - Optional. The exchanger of the MX record (regular expression).
  pref      - Optional. The Preference of the MX record (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS MX objects that match "domain.com" in the default DNS view
    my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::MX",
    name   => 'domain\.com',
    view   => "default" );

# search for all DNS MX objects in the domain.com zone of the default DNS view
    my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::MX",
    zone   => "domain.com",
    view   => "default" );

# search all DNS MX records with a given extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::Record::MX",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an MX record

comment( )

Use this method to add or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for this record in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting comment
my $comment = $bindmx->comment();
# Modifying comment
$bindmx->comment("add mx2.domain.com");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting disable
my $disable = $bindmx->disable()
# Modifying disable
$bindmx->disable("true");
```

exchanger( )

Use this method to set or retrieve the exchanger.

The name of the mail exchanger in Fully Qualified Domain Name (FQDN) format.

Parameter

Text with the name of the mail exchanger for the DNS MX record.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting exchanger
my $exchanger = $bindmx->exchanger();
# Modifying exchanger
$bindmx->exchanger("exchanger2.domain.com");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS MX record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $bindmx->extensible_attributes();
#Modify extensible attributes
$bindmx->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the domain name.

The name of the DNS MX object in Fully Qualified Domain Name (FQDN) format. If you want to define an MX record for a domain whose name matches the zone in which you define the MX record, leave this field empty. If you want to define an MX record for a subdomain, enter that subdomain name here. If you want to define an MX record for a domain and all its subdomains, enter an asterisk (*) here to create a wildcard MX record.

Parameter

Text with the name for the DNS MX record.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting name
my $name = $bindmx->name();
# Modifying name
```

```
$bindmx->name("mx2.domain.com");
```

pref( )

Use this method to set or retrieve the Preference value.

The preference value of the DNS MX object. Preference value of MX record for the zone range from 0 to 65535. Low values are more preferred.

Parameter

Enter an unsigned integer between 0-65535. The priority determines the order in which a client attempts to contact the target mail exchanger. The highest priority is 0 and is queried first.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting pref
my $pref = $bindmx->pref();
# Modifying pref
$bindmx->pref(3);
```

ttl( )

Use this method to configure or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $bindmx->ttl();
#Modify ttl
$bindmx->ttl(1800);
#Un-override ttl
$bindmx->ttl(undef);
```

views( )

Use this method to specify or retrieve the view of the MX record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the 'default' view, which means the MX record is located under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $bindmx->views();
#Modify views, list of Infoblox::DNS::View objects
$bindmx->views([$view1]);
```

zone( )

Use this method to retrieve the zone name of a DNS MX record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $bindmx->zone();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS MX record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the zone prior to MX  record insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
   die("Construct zone failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS MX record

```
#Construct a DNS MX object
 my $bindmx = Infoblox::DNS::Record::MX->new(
   name      => "bind_mx.domain.com",
   comment   => "add mx bind_mx.domain.com",
   pref      => 1,
   exchanger => "exchanger.domain.com",
 );

unless ($bindmx) {
   die("Construct DNS record MX failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
}
print "DNS MX object created successfully\n"; #Add the DNS MX record object to Infoblox appliance through
a session
$session->add($bindmx)
    or die("Add record MX failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS MX object added to server successfully\n";
```

#Search for a specific DNS MX record

```
#Search all MX records that match "domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::MX",
    name   => 'domain\.com'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS MX object found at least 1 matching entry\n";

#Search all MX records that start with "bind" and end with "domain.com"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::MX",
    name   => '^bind.*\.domain\.com\$'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS MX object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS MX record

```
#Get MX record through the session
my  @retrieved_objs = $session->get(
    object    => "Infoblox::DNS::Record::MX",
    name      => "bind_mx.domain.com",
    pref      => 1,
    exchanger => "exchanger.domain.com",
);

my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS MX object found at least 1 matching entry\n";

#Modify one of the attributes of the specified MX record
$object->exchanger("exc.domain.com");

#Apply the changes
$session->modify($object)
    or die("Modify record MX failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS MX object modified successfully \n";
```

#Remove a DNS MX record

```
 #Get MX record through the session
 my @retrieved_objs = $session->get(
     object => "Infoblox::DNS::Record::MX",
     name   => "bind_mx.domain.com"
  );
   my $object = $retrieved_objs[0];
   unless ($object) {
    die("Get record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
  }
  print "Get DNS MX object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove record MX failed: ",
```

```
        $session->status_code() . ":" . $session->status_detail());
print "DNS MX object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::NAPTR - NAPTR record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - flags( )
  - name( )
  - order( )
  - preference( )
  - regexp( )
  - replacement( )
  - services( )
  - ttl( )
  - views( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::NAPTR - NAPTR record object.

DESCRIPTION

A DNS NAPTR object represents a Naming Authority Pointer (NAPTR) resource record. This resource record specifies a regular expression-based rewrite rule that, when applied to an existing string, produces a new domain name or URI.

CONSTRUCTOR

```
 my $naptr = Infoblox::DNS::Record::NAPTR->new(
     name            => $fqdn,            #Required
     order           => $num,             #Required
     preference      => $num,             #Required
     flags           => $string,          #Optional
     services        => $string,          #Optional
     regexp          => $string,          #Optional
     replacement     => $string           #Required
     comment         => $string,          #Optional / Default is empty
     disable         => "true" | "false", #Optional / Default is "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl             => $num | undef,     #Optional / Default is undefined
     views           => [$View1],         #Optional / Default is "default" view
 );
```

Note that you must specify only one DNS view for the attribute "views".

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that can be applied to an NAPTR record object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $naptr = Infoblox::DNS::Record::NAPTR->new(
    name            => 'naptr.domain.com',
    order           => 100,
    preference      => 10,
    flags           => 'U',
    services        => 'http+E2U',
    regexp          => '!http://my[.](.*)!1!i',
    replacement     => 'domain2.com',
    comment         => 'Sample NAPTR record',
    disable         => 'false',
    ttl             => 84600,
    views           => [$view1],
);
# Submit for addition
my $response = $session->add( $naptr );
```

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get specific NAPTR record objects:

    comment         - Optional. Descriptive comment in string format
    extensible_attributes - Optional. A hash reference containing extensible attributes.
    flags           - Optional. Flags in string format
    name            - Optional. FQDN of the record in string format
    order           - Optional. Order parameter (unsigned integer)
    preference      - Optional. Preference parameter (unsigned integer)
    replacement     - Optional. Replacement field in string format
    services        - Optional. Services field in string format
    view            - Optional. The DNS view in which the NAPTR record is located. By default, all
DNS views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
    zone            - Optional. A zone name in FQDN format.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::NAPTR",
    name   => "naptr1.domain.com",
    view   => "default" );

#get all DNS A records with the extensible attribute 'Site'
my @retrieved_objs = $session->get(
   object => "Infoblox::DNS::Record::NAPTR",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment.
$naptr->comment("this is a modified comment");
# Submit the modification
my $response = $session->modify( $naptr );
```

Infoblox::Session->remove( )

Use this method to submit the removal of an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the object, and then submit it for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::NAPTR",
    name   => "naptr1.domain.com",
    view   => "default" );
# Find the desired object in the retrieved list.
my $desired_rec = $retrieved_objs[0];
# Submit for removal
```

```
my $response = $session->remove( $desired_rec );
```

Infoblox::Session->search( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to search the NAPTR record objects:

    comment         - Optional. Descriptive comment (regular expression)
    extensible_attributes - Optional. A hash reference containing extensible attributes.
    flags           - Optional. Flags in string format (regular expression)
    name            - Optional. FQDN of the record in string format (regular expression)
    order           - Optional. Order parameter (unsigned integer, exact search)
    preference      - Optional. Preference parameter (unsigned integer, exact search)
    replacement     - Optional. Replacement field in string format (regular expression)
    services        - Optional. Services field in string format (regular expression)
    view            - Optional. The DNS view in which the NAPTR record is located. By default, all
DNS views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only. Only exact search is supported.
    zone            - Optional. A zone name in FQDN format (exact search)
```

Example

```
# search for all NAPTR objects that match "domain.com" in the default DNS view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::NAPTR",
    name   => '.*domain\.com',
    view   => "default" );

# search for all NAPTR records in the "domain.com" zone of the default DNS view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::NAPTR",
    zone   => "domain.com",
    view   => "default" );

#get all NAPTR records with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
  object => "Infoblox::DNS::Record::NAPTR",
  extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of an NAPTR record object.

comment( )

Use this method to set or retrieve the descriptive comment of an NAPTR record object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the descriptive comment
my $comment = $naptr->comment();
#Modify the comment
$naptr->comment("This is a new comment");
```

disable( )

Use this method to set or retrieve the disable flag of an NAPTR record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false, which indicates that the NAPTR record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the disable field
my $disable = $naptr->disable();
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with an NAPTR record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get  extensible attributes
my $ref_extensible_attributes = $naptr->extensible_attributes();
#Modify extensible attributes
$naptr->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

flags( )

Use this method to retrieve the flags to control the interpretation of the fields of an NAPTR record object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Currently supported values for the flags field are "U", "S", "P" and "A".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the flags field
my $flags = $naptr->flags();
#Modify flags
$naptr->flags("U");
```

name( )

Use this method to retrieve the FQDN (Fully Qualified Domain Name) of the NAPTR record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name parameter is a string in FQDN format. The FQDN consists of the record name followed by the domain name (example: rec.abc.com). A record name can have a maximum of 256 bytes.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the FQDN of the record
my $name = $naptr->name();
#Modify the FQDN of the record
$naptr->name("rec2.domain.com");
```

order( )

Use this method to retrieve the order parameter of an NAPTR record. This parameter specifies the order in which the NAPTR rules are applied when multiple rules are present.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The order parameter is a 16-bit unsigned integer.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the order parameter
my $order = $naptr->order();
#Modify the order
$naptr->order(200);
```

preference( )

Use this method to retrieve the preference field of an NAPTR record. The preference field determines the order NAPTR records are processed when multiple records with the same order parameter are present.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The preference parameter is a 16-bit integer.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the preference attribute value
my $preference = $naptr->preference();
#Modify preference
$naptr->preference(300);
```

regexp( )

Use this method to specify the regular expression-based rewriting rule of an NAPTR record. This should be a POSIX compliant regular expression, including the substitution rule and flags. Refer to RFC 2915 for the field syntax details.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid parameter is a POSIX compliant regular expression or substitution expression.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get regexp
```

```
my $regexp = $naptr->regexp();
#Modify regexp
$naptr->regexp("!http://(.*)!1!i");
```

replacement( )

Use this method to retrieve the replacement field of an NAPTR record object. For nonterminal NAPTR records, this field specifies the next domain name to look up.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The replacement field is in FQDN format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the replacement field
my $replacement = $naptr->replacement();
#Modify the replacement field
$naptr->replacement("__tcp__.domain.com");
```

services( )

Use this field to specify services. The services field contains protocol and service identifiers, for example "http+E2U" or "SIPS+D2T".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The services field in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get services
my $services = $naptr->services();
#Modify services
$naptr->services("http+E2U");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined, which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (from 0 to 4294967295) that represents the duration, in seconds, that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the TTL of the record
my $ttl = $naptr->ttl();
#Modify TTL
$naptr->ttl(900);
#Use zone ttl settings
```

```
$naptr->ttl(undef);
```

views( )

Use this method to set or retrieve the DNS view of the NAPTR record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" view, which means the NAPTR record is located in the default DNS view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $naptr->views();
#Modify views, list of Infoblox::DNS::View objects
$naptr->views([$view1]);
```

zone( )

Use this method to retrieve the zone name of an NAPTR record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $naptr->zone();
```

SAMPLE CODE

The following sample code demonstrates the session methods on an NAPTR record object.

#Preparation prior to a NAPTR record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

B<#Create a session to the Infoblox appliance>

 my $session = Infoblox::Session->new(
     master   => $host_ip,
     username => "admin",
     password => "infoblox"
 );
 unless ($session) {
    die("Construct session failed: ",
        $session->status_code() . ":" . $session->status_detail());
 }
 print "Session created successfully\n";

 #Create the zone prior to an A record insertion
 my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
 unless ($zone) {
    die("Construct zone failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
 }
 print "Zone object created successfully\n";
```

```
#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS NAPTR record

```
#Construct a DNS NAPTR object
my $bind_naptr = Infoblox::DNS::Record::NAPTR->new(
    name     => "bind_naptr.domain.com",
    comment  => " this is a demo bind_naptr record ",
    services => "http+E2U",
    flags    => "U",
    regexp   => "!http://(.*)!1!i",
    preference => 100,
    order    => 10,
    replacement => '_http._tcp.domain.com',
);
unless ($bind_naptr) {
   die("Construct DNS NAPTR record failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS NAPTR record object created successfully\n";

#Add the DNS NAPTR record object to the Infoblox appliance through a session
$session->add($bind_naptr)
    or die("Adding NAPTR record failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS NAPTR object added to server successfully\n";
```

#Search for a specific DNS NAPTR record

```
#Search all NAPTR records in the zone
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::NAPTR",
    name   => '.*\.domain\.com'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search for NAPTR record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS NAPTR record object found at least 1 matching entry\n";

#Search for all NAPTR records that start with "bind" in the zone
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::NAPTR",
    name   => 'bind.*\.domain\.com'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search for NAPTR record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Searching DNS NAPTR object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS NAPTR record

```
#Get a record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::NAPTR",
    name   => "bind_naptr.domain.com"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get NAPTR record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS NAPTR object found at least 1 matching entry\n";

#Modify some attributes of the specified NAPTR record
$object->preference(200);
```

```
$object->order(100);

#Apply the changes
$session->modify($object)
    or die("Modify NAPTR record failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS NAPTR object modified successfully \n";
```

#Remove a DNS NAPTR record

```
#Get NAPTR record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::NAPTR",
    name   => "bind_naptr.domain.com"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get NAPTR record failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get NAPTR DNS object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove NAPTR record failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "DNS NAPTR object removed successfully \n";

#Remove the zone

#Get Zone object through the session
@retrieved_objs = $session->get(
    object => "Infoblox::DNS::Zone",
    name   => "domain.com"
);
$object = $retrieved_objs[0];
unless ($object) {
    die("Get zone failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get zone object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove zone object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Zone object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(),Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::DNS::Zone

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::NS - DNS NS record object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - nameserver( )
  - addresses( )
  - views( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::NS - DNS NS record object

DESCRIPTION

An NS record identifies an authoritative DNS server for a domain. Each authoritative DNS server must have an NS record. The appliance automatically creates an NS record when you assign a grid member as the primary server for a zone. You can manually create NS records for other zones.

Note that you must specify only one DNS view for the attribute "views".

CONSTRUCTOR

```
my $ns = Infoblox::DNS::Record::NS->new(
    name                => $name,                   #Required
    nameserver          => $nameserver,             #Required
    addresses           => [$ns1, $ns2, ...],       #Required
    views               => [$View1]                 #Optional / Default is "default" view
);
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS NS Record object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $nsrecord = Infoblox::DNS::Record::NS->new(
    nameserver => "ns1.server.isp.org",
    name       => "domain.com",
    addresses  => [$ns1, $ns2],
);
# Submit for addition
my $response = $session->add( $nsrecord );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS NS Record object:

 name        - Optional.  A domain name in FQDN (Fully Qualified Domain Name) format.
 nameserver  - Optional.  A domain name in FQDN (Fully Qualified Domain Name) format.
 view        - Optional.  The DNS view in which the NS record is located. By default, all DNS views
are searched.
```

Example

```
my @retrieved_objs = $session->get(
   object    => "Infoblox::DNS::Record::NS",
   name      => "domain.com",
   view      => "default" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the addresses.
$nsrecord->addresses([$ns3]);
# Submit modification
my $response = $session->modify( $nsrecord );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my @retrieved_objs = $session->get(
    object    => "Infoblox::DNS::Record::NS",
    name      => "domain.com",
    view      => "default" );
# Find the desired object from the retrieved list.
my $desired_ns = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_ns );
```

Infoblox::Session->search( )

Use this method to search for DNS NS Record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS NS Record object:

 name          - Optional. A domain name in FQDN (Fully Qualified Domain Name) format (regular
expression).
 nameserver    - Optional. A domain name in FQDN (Fully Qualified Domain Name) format (regular
expression).
 view          - Optional. The DNS view in which the NS record is located. By default, all DNS views
are searched.
```

Example

```
# search for all DNS NS objects that match "domain.com" in the default DNS view
    my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::NS",
    name   => 'domain\.com',
    view   => "default" );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an NS record.

name( )

Use this method to set or retrieve the domain name of the zone to be redirected.

Parameter

A domain name in FQDN format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting name
my $name = $nsrecord->name();
# Modifying name
$nsrecord->name("domain.com");
```

nameserver( )

Use this method to set or retrieve the domain name of an authoritative server for the redirected zone.

Parameter

A domain name in FQDN format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting nameserver
my $nameserver = $nsrecord->nameserver();
# Modifying nameserver
$nsrecord->nameserver("ns1.server.isp.org");
```

addresses( )

Use this method to set or retrieve the IP addresses of the name servers associated with the NS record.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Nameserver::Address objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting addresses
my $nslist = $nsrecord->addresses();
# Modifying addresses
$nsrecord->addresses([$ns1, $ns2]);
```

views( )

Use this method to specify or retrieve the DNS view of the NS record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the 'default' view, which means the NS record is associated with a zone in the system-defined default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
```

```
my $ref_views = $nsrecord->views();
#Modify views, list of Infoblox::DNS::View objects
$nsrecord->views([$view1]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS NS record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the zone prior to NS record insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
   die("Construct zone failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS NS record

```
my $ns1 = Infoblox::DNS::Nameserver::Address->new(
                                                        address  => '10.0.0.1',
                                                auto_create_ptr => 'false'
                                                );

my $ns2 = Infoblox::DNS::Nameserver::Address->new(
                                                address         => '10.0.0.2',
                                                );

 #Construct a DNS NS object
  my $nsrecord = Infoblox::DNS::Record::NS->new(
    nameserver => "ns1.server.isp.org",
    name       => "domain.com",
    addresses  => [$ns1, $ns2],
  );

unless ($nsrecord) {
   die("Construct DNS record NS failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS NS object created successfully\n"; #Add the DNS NS record object to Infoblox appliance through
a session
$session->add($nsrecord)
    or die("Add record NS failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS NS object added to server successfully\n";
```

#Search for a specific DNS NS record

```
#Search all NS records that match "domain.com"
my @retrieved_objs = $session->search(
    object     => "Infoblox::DNS::Record::NS",
    nameserver => "ns1.server.isp.org",
    name       => 'domain\.com'
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record NS failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS NS object found at least 1 matching entry\n";
```

#Get and modify a DNS NS record

```
#Get NS record through the session
my  @retrieved_objs = $session->get(
    object     => "Infoblox::DNS::Record::NS",
    nameserver => "ns1.server.isp.org",
    name       => "domain.com",
);

my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record NS failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS NS object found at least 1 matching entry\n";

#Modify one of the attributes of the specified NS record
$object->addresses([$ns1]);

#Apply the changes
$session->modify($object)
    or die("Modify record NS failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS NS object modified successfully \n";
```

#Remove a DNS NS record

```
 #Get NS record through the session
 my @retrieved_objs = $session->get(
     object     => "Infoblox::DNS::Record::NS",
     nameserver => "ns1.server.isp.org",
     name       => "domain.com"
  );
   my $object = $retrieved_objs[0];
   unless ($object) {
    die("Get record NS failed: ",
        $session->status_code() . ":" . $session->status_detail());
  }
  print "Get DNS NS object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove record NS failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "DNS NS object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::NSEC - NSEC Record object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - next_domain_name( )
  - ttl( )
  - types( )
  - view( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::NSEC - NSEC Record object

DESCRIPTION

The NSEC resource record is one of the resource records included in the DNS security extension mechanism (DNSSEC). This record is used to provide authenticated denial of existence of a resource record in responce to a resolver query.

The NSEC record is defined in RFC 4034.

The NSEC records are automatically generated upon the signing of an authoritative zone. The NSEC Record object is read-only.

CONSTRUCTOR

The Infoblox::DNS::Record::NSEC object is a read-only object and does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to an NSEC Record object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
 Apply the following attributes to get the NSEC Record object(s):

     name            - Optional. FQDN of the NSEC record (.e., FQDN of the owning RRset).
     next_domain_name - Optional. Next owner name (in the canonical ordering of the zone) that has
authoritative data
     ttl             - Optional. TTL value for the record
     types           - Optional. Identifies the RRset types that exists at the NSEC RR's owner name
     view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
     zone            - Optional. A zone name in FQDN format.
```

Example

```
#get an NSEC record by name, DNS view name and record type
my $nsec=$session->get(
                       object => "Infoblox::DNS::Record::NSEC",
                       name   => "a_record.test.com",
                       types  => "A",
                       view   => "default"
                     );
```

Infoblox::Session->search( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to search the NSEC Record object(s):

    name            - Optional. FQDN of the NSEC record. Regular expression.
    next_domain_name - Optional. Next owner name (in the canonical ordering of the zone) that has
authoritative data (regular expression).
    ttl             - Optional. TTL value for the record
    types           - Optional. Identifies the RRset types that exist at the NSEC RR's owner name
(regular expression)
    view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
    zone            - Optional. A zone name in FQDN format.
```

Example

```
  #search for all NSEC records that match the regular expression for name and record type
  my @retrieved_objs=$session->search(
                        object => "Infoblox::DNS::Record::NSEC",
                        name   => "rec.*[.]test[.]com",
                        types  => "A|MX",
                      );

# search for all NSEC records in the "domain.com" zone of the default DNS view
# Note that the 'default' view is assumed implicitly here
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::NSEC",
    zone   => "domain.com",
   );
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of an NSEC Record object.

name( )

Use this method to retrieve the FQDN of the NSEC record (which is the same as the name of owning record).

Parameter

none

Returns

The method returns the FQDN of the NSEC record attribute value.

Example

```
#Get the FQDN of the NSEC record
my $name = $nsec->name();
```

next_domain_name( )

Use this method to retrieve the next owner name, in the canonical order of the zone, that has authoritative data. Is it the canonical order of the domain names?

Parameter

none

Returns

The method returns the next owner name (in the canonical ordering of the zone) that has authoritative data. The returned data is a string in FQDN format.

Example

```
#Get  next owner name (in the canonical ordering of the zone) that has authoritative data
my $next_domain_name = $nsec->next_domain_name();
```

ttl( )

Use this method to retrieve the Time to Live (TTL) value of an NSEC record object.

Parameter

none

Returns

The method returns the TTL attribute value. The returned parameter is a 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the record is cached. Zero indicates that the record should not be cached.

Example

```
#Get TTL
my $ttl = $nsec->ttl();
```

types( )

Use this method to retrieve the types field of an NSEC record. This field identifies the RRSet types that exist at the original owner name of the NSEC RR.

Parameter

none

Returns

The method returns the RRSet types that exist at the owner name of the NSEC RR attribute value. The returned value is a string containing RR type mnemonics separated by a space; for example, "NS SOA RRSIG DNSKEY".

Example

```
#Get the types field value
my $types = $nsec->types();
```

view( )

Use this method to retrieve the DNS view object that contains the NSEC Record object.

Parameter

none

Returns

The method returns the Infoblox::DNS::View object that contains the NSEC record.

Example

```
#Get the view
my $view = $nsec->view();
```

zone( )

Use this method to retrieve the zone name of an NSEC record.

Parameter

none

Returns

Returns the zone name in FQDN format.

Example

```
# Get zone
my $zone = $nsec->zone();
```

SAMPLE CODE

The following sample code demonstrates the session methods on an NSEC record object.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
       my $session = Infoblox::Session->new(
```

```
        master   => "192.168.1.2",
        username => "admin",
        password => "infoblox"
    );
    unless ($session) {
       die("Construct session failed: ",
           $session->status_code() . ":" . $session->status_detail());
    }
    print "Session created successfully\n";

    #Enable DNSSEC in the default view

    my $default_view=$session->get(
                                   object=> "Infoblox::DNS::View",
                                   name => "default"
                                  );
    unless($default_view) {
        die("Getting the default view failed:",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
    }
    print "Got the default view successfully\n";

    $default_view->dnssec_enabled("true")
        or die("Changing the dnssec_enabled in the default view failed:",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
    $session->modify($default_view)
        or die("Changing the dnssec_enabled in the default view failed:",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
```

#Create a signed zone and populate it with an A record to demonstrate NSEC record

```
    print "Creating Member primary server for the zone\n";
    my $primary=Infoblox::DNS::Member->new(
                                           ipv4addr => "192.168.1.2",
                                           name => "infoblox.localdomain",
                                          );
    unless($primary) {
       die("Unable to create primary server object: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
    }

    my $zone = Infoblox::DNS::Zone->new(
                                        name => "domain.com",
                                        primary => $primary
                                       );
    unless ($zone) {
       die("Construct zone failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
    }
    print "Zone object created successfully\n";

    #Verify if the zone exists
    my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
    unless ($object) {
       print "Zone does not exist on server, safe to add the zone\n";
       $session->add($zone)
          or die("Add zone failed: ",
                 $session->status_code() . ":" . $session->status_detail());
    }
    print "Zone added successfully\n";

    #Adding A record to the zone to demonstrate the corresponding NSEC record
    my $a_record=Infoblox::DNS::Record::A->new(
                                               ipv4addr => "10.9.8.7",
                                               name => "recorda.domain.com"
                                              );
    unless($a_record) {
       die("Creating A record failed : ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
    }

    $object = $session->get( object => "Infoblox::DNS::Record::A", name => "recorda.domain.com" );
    unless ($object) {
       print "A record does not exist on the server, safe to add the A record\n";
       $session->add($a_record)
         or die("Adding A record failed: ",
```

```
                    Infoblox::status_code() . ":" . Infoblox::status_detail());
        }
        print "A record added to the zone successfully\n";

        #Retrieving zone back from the server in order to sign it
        $zone = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
        unless($zone) {
           die("Retrieving zone back failed: ",
                Infoblox::status_code( ). ":". Infoblox::status_detail( ));
        }
        print "Zone retrieved for signing successfully.\n";

        $zone->dnssec_ksk_algorithm("RSASHA1") &&
        $zone->dnssec_zsk_algorithm("RSASHA1") &&
        $zone->dnssec_ksk_size(640) &&
        $zone->dnssec_zsk_size(640)
          or die("Changing the zone DNSSEC setting failed: ",
                 Infoblox::status_code() . ":" . Infoblox::status_detail());
        $session->modify($zone)
          or die("Modifying dnssec values in zone failed: ",
                 Infoblox::status_code() . ":" . Infoblox::status_detail());
        print "Zone modified successfully\n";

        #Signing the zone
        $zone->dnssec_signed("true")
           or die("Signing of the zone failed: ",
                   Infoblox::status_code() . ":" . Infoblox::status_detail());
               print "Zone signed successfully\n";

#Getting the NSEC record corresponding to the A record

        my $nsec_record_a=$session->get(
                                  object => "Infoblox::DNS::Record::NSEC",
                                  name   => "recorda.domain.com",
                                  view   => "default"
                                 );
             unless($nsec_record_a) {
         die("Getting NSEC corresponding to the A record failed: ",
             Infoblox::status_code() . ":" . Infoblox::status_detail());
             }
             print "Got NSEC record successfully, next_domain_name value: ".$nsec_record_a-
>next_domain_name()."\n";

#Searching for NSEC objects using regular expressions

       my @retrieved_objs=$session->search(
                                        object => "Infoblox::DNS::Record::NSEC",
                                        name   => ".*domain[.]com",
                                        types  => "A|MX|SOA",
                                        view   => "default"
                                      );
           unless(@retrieved_objs>0) {
         die("Searching for NSEC objects failed: ",
             Infoblox::status_code() . ":" . Infoblox::status_detail());
           }
           print "Search for the NSEC objects successful, ".scalar(@retrieved_objs)." objects found\n";

#Removing the created zone and cleaning up the view

       $session->remove($zone)
         or die("Unable to remove the zone: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
       print "Zone removed successfully\n";

       $default_view->dnssec_enabled("false")
       && $default_view->override_dnssec("false")
       && $session->modify($default_view)
        or die("Restoring dnssec_enabled value in the default view failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::DNS::Record::NSEC3, Infoblox::DNS::Record::NSEC3PARAM, Infoblox::DNS::View, Infoblox::DNS::Zone

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::NSEC3 - NSEC3 Record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
- METHODS
  - flags( )
  - hash_algorithm( )
  - iterations( )
  - name( )
  - next_hash_owner( )
  - salt( )
  - ttl( )
  - types( )
  - view( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::NSEC3 - NSEC3 Record object.

DESCRIPTION

The NSEC3 resource record is a part of DNS security extension (DNSSEC) authentication mechanism. It is used for authenticated denial of existence of DNS Resource record sets in the DNSSEC signed authoritative zones. The functionality that NSEC3 records provide is similar to the functionality of NSEC records, but NSEC3 includes cryptographic protection against zone enumeration.

The NSEC3 resource record is desribed in RFC 5155.

All the NSEC3 records are automatically generated during signing of the corresponding zone.

CONSTRUCTOR

The Infoblox::DNS::Record::NSEC3 object is a read-only object and does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to an NSEC3 Record object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
 Apply the following attributes to get the NSEC3 Record object(s):

     flags           - Optional. Integer containing 8 one-bit flags that can be used to indicate
different processing.
     hash_algorithm  - Optional. Integer representing the cryptographic hash algorithm used to
construct the hash-value.
     iterations      - Optional. Number of additional times the hash function has been performed.
     name            - Optional. FQDN of the NSEC3 record (contains zone name and the hashed record
name).
     next_hash_owner - Optional. Next hashed owner name in hash order.
     salt            - Optional. Salt value.
     ttl             - Optional. TTL value.
     types           - Optional. Identifies the RRSet types that exist at the original owner name of
the NSEC3 RR.
     view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
     zone            - Optional. A zone name in FQDN format.
```

Example

```
# get all the NSEC3 records corresponding to 'A' records in the default DNS view.
my @retrieved_objs = $session->get(
                     object => "Infoblox::DNS::Record::NSEC3",
                     types  => "A",
                     view   => "default" );
```

Infoblox::Session->search( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to search the NSEC3 Record object(s):

     flags           - Optional. Integer containing 8 one-bit flags that can be used to indicate
different processing.
     hash_algorithm  - Optional. Integer representing the cryptographic hash algorithm used to
construct the hash-value.
     iterations      - Optional. Number of additional times the hash function has been performed.
     name            - Optional. FQDN of the NSEC3 record (contains zone name and the hashed record
name). Regular expression
     next_hash_owner - Optional. Next hashed owner name in hash order (regular expression).
     salt            - Optional. Salt value (regular expression).
     ttl             - Optional. TTL value.
     types           - Optional. Identifies the RRSet types that exist at the original owner name of
the NSEC3 RR (regular expression).
     view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
     zone            - Optional. A zone name in FQDN format.
```

Example

```
# get all the NSEC3 records in the zone test.com.
my @retrieved_objs = $session->get(
                     object => "Infoblox::DNS::Record::NSEC3",
                     name   => ".*test[.]com",
                     );

# search for all NSEC3 records in the "domain.com" zone of the default DNS view
# Note that the 'default' view is assumed implicitly here
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::NSEC3",
    zone   => "domain.com",
   );
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of an NSEC3 Record object.

flags( )

Use this method to retrieve the flags field of an NSEC3 Record object.

Parameter

none

Returns

The method returns the integer that contains 8 one-bit flags that can be used to indicate different processing. Currently only the least significant bit of the flags field is meaningful. It represents the opt-out flag, which is used to determine the processing of non-secure delegations.

Example

```
#Get the flags field.
my $flags = $nsec3->flags();
```

hash_algorithm( )

Use this method to retrieve the cryptographic hash algorithm used to construct the hash value of an NSEC3 Record object.

Parameter

none

Returns

The method returns an integer that corresponds to the cryptographic hash algorithm used to construct the hash value.

Example

```
#Get hash algorithm
my $hash_algorithm = $nsec3->hash_algorithm();
```

iterations( )

Use this method to retrieve the number of iterations of the hash function performed to obtain the hashed name.

Parameter

none

Returns

The method returns the number of additional times the hash function has been performed to compute the hash value. This is an integer in the range 0 through 65535, inclusive.

Example

```
#Get the iterations parameter
my $iterations = $nsec3->iterations();
```

name( )

Use this method to retrieve the name of NSEC3 record in FQDN format. The name of NSEC3 record consists of the hashed owner name prepended to the name of the zone, containing NSEC3 record.

Parameter

none

Returns

The method returns the name of NSEC3 record.

Example

```
#Get the NSEC3 record name
my $name = $nsec3->name();
```

next_hash_owner( )

Use this method to retrieve the next hashed owner name. This is a hash of an owner name that immediately follows the owner name of the given NSEC3 record in the hash order.

Parameter

none

Returns

The method returns the next hashed owner name. It is a string in base-32 encoding.

Example

```
#Get the next hashed owner name
my $next_hash_owner = $nsec3->next_hash_owner();
```

salt( )

Use this method to retrieve the salt value of a NSEC3 record. The salt is prepended to the original owner name before hashing and on each of the hash iterations in order to defend against pre-calculated dictionary attacks.

Parameter

none

Returns

The method returns a salt value. The returned value is a string containing a sequence of hexadecimal digits representing the salt or '-' if the salt is not used.

Example

```
#Get the salt
my $salt = $nsec3->salt();
```

ttl( )

Use this method to retrieve the Time to Live (TTL) value of a NSEC3 Record object.

Parameter

none

Returns

The method returns the TTL attribute value. The returned parameter is a 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Example

```
#Get TTL
my $ttl = $nsec3->ttl();
```

types( )

Use this method to retrieve the types field of an NSEC3 record. Types field identifies the RRSet types that exist at the original owner name of the NSEC3 RR.

Parameter

none

Returns

The method returns the RRSet types that exist at the original owner name of the NSEC3 RR attribute value. The returned value is a string containing space separated RR types mnemonics, e.g. "NS SOA RRSIG DNSKEY NSEC3PARAM".

Example

```
#Get the types field value
my $types = $nsec3->types();
```

view( )

Use this method to retrieve the DNS view object that contains the NSEC3 Record object.

Parameter

none

Returns

The method returns the Infoblox::DNS::View object that contains the NSEC3 record.

Example

```
#Get the DNS view
my $view = $nsec3->view();
```

zone( )

Use this method to retrieve the zone name of a NSEC3 record.

Parameter

none

Returns

Returns the zone name of the zone that contains the NSEC3 record. The zone name is in FQDN format.

Example

```
# Get zone
my $zone = $nsec3->zone();
```

SAMPLE CODE

The following sample code demonstrates the session methods on a NSEC3 Record object.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device
       my $session = Infoblox::Session->new(
           master   => "192.168.1.2",
           username => "admin",
           password => "infoblox"
       );
       unless ($session) {
          die("Construct session failed: ",
              $session->status_code() . ":" . $session->status_detail());
       }
       print "Session created successfully\n";

       #Enable DNSSEC in the default view

       my $default_view=$session->get(
                                   object=> "Infoblox::DNS::View",
                                   name => "default"
                                  );
       unless($default_view) {
           die("Getting the default view failed:",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
       print "Got the default view successfully\n";

       $default_view->dnssec_enabled("true")
           or die("Changing the dnssec_enabled in the default view failed:",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
       $session->modify($default_view)
           or die("Changing the dnssec_enabled in the default view failed:",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());

#Create a signed zone and populate it with an A record

       print "Creating Member primary server for the zone\n";
       my $primary=Infoblox::DNS::Member->new(
                                          ipv4addr => "192.168.1.2",
                                          name => "infoblox.localdomain",
                                         );
       unless($primary) {
          die("Unable to create primary server object: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
       }

       my $zone = Infoblox::DNS::Zone->new(
                                        name => "domain.com",
                                        primary => $primary
                                       );
       unless ($zone) {
          die("Construct zone failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
       print "Zone object created successfully\n";

       #Verify if the zone exists
       my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
       unless ($object) {
          print "Zone does not exist on server, safe to add the zone\n";
          $session->add($zone)
             or die("Add zone failed: ",
                    $session->status_code() . ":" . $session->status_detail());
       }
       print "Zone added successfully\n";

       #Adding A record to the zone to demonstrate the corresponding NSEC3 record
       my $a_record=Infoblox::DNS::Record::A->new(
                                              ipv4addr => "10.9.8.7",
                                              name => "recorda.domain.com"
                                             );
       unless($a_record) {
          die("Creating A record failed : ",
             Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
}

$object = $session->get( object => "Infoblox::DNS::Record::A", name => "recorda.domain.com" );
unless ($object) {
   print "A record does not exist on the server, safe to add the A record\n";
   $session->add($a_record)
     or die("Adding A record failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "A record added to the zone successfully\n";

#Retrieving zone back from the server in order to sign it
$zone = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless($zone) {
   die("Retrieving zone back failed: ",
        Infoblox::status_code( ). ":". Infoblox::status_detail( ));
}
print "Zone retrieved for signing successfully.\n";

$zone->dnssec_ksk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_zsk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_ksk_size(640) &&
$zone->dnssec_zsk_size(640)
  or die("Changing the zone DNSSEC setting failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->modify($zone)
  or die("Modifying dnssec values in zone failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Zone modified successfully\n";

#Signing the zone
$zone->dnssec_signed("true")
   or die("Signing of the zone failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
       print "Zone signed successfully\n";
```

#Getting the NSEC3 record corresponding to the A record

```
my $nsec3_record_a=$session->get(
                            object => "Infoblox::DNS::Record::NSEC3",
                            types     => "A RRSIG",
                            view   => "default"
                           );
     unless($nsec3_record_a) {
  die("Getting NSEC3 corresponding to the A record failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
     }
     print "Got NSEC3 record successfully, name value: ".$nsec3_record_a->name()."\n";
```

#Searching for NSEC3 objects using regular expressions

```
my @retrieved_objs=$session->search(
                                  object => "Infoblox::DNS::Record::NSEC3",
                                  name   => ".*domain[.]com",
                                  types  => "A|MX|SOA",
                                  view   => "default"
                                 );
   unless(@retrieved_objs>0) {
  die("Searching for NSEC3 objects failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
   }
   print "Search for the NSEC3 objects successful, ".scalar(@retrieved_objs)." objects found\n";
```

#Removing the created zone and cleaning up the view

```
$session->remove($zone)
  or die("Unable to remove the zone: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Zone removed successfully\n";

$default_view->dnssec_enabled("false")
&& $default_view->override_dnssec("false")
&& $session->modify($default_view)
 or die("Restoring dnssec_enabled value in the default view failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::DNS::Record::NSEC, Infoblox::DNS::Record::NSEC3PARAM, Infoblox::DNS::View, Infoblox::DNS::Zone

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::NSEC3PARAM - NSEC3PARAM Record object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
- METHODS
  - flags( )
  - hash_algorithm( )
  - iterations( )
  - name( )
  - salt( )
  - ttl( )
  - view( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::NSEC3PARAM - NSEC3PARAM Record object

DESCRIPTION

The NSEC3PARAM resource record is one of the resource records of the DNS security extension (DNSSEC) authentication mechanism. The NSEC3PARAM RR contains the NSEC3 parameters (hash algorithm, flags, iterations, and salt) needed by authoritative servers to calculate hashed owner names for NSEC3 records. The NSEC3PARAM record is present at the apex of the DNSSEC-signed authoritative zone.

The NSEC3PARAM resource record is desribed in RFC 5155.

The NSEC3PARAM record is generated automatically upon the signing of the corresponding zone. It is a read-only object.

CONSTRUCTOR

The Infoblox::DNS::NSEC3PARAM object is a read-only object and does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to an NSEC3PARAM Record object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
 Apply the following attributes to get the NSEC3PARAM Record object(s):

     flags           - Optional. Integer containing 8 one-bit flags that can be used to indicate
different processing.
     hash_algorithm  - Optional. Integer representing the cryptographic hash algorithm.
     iterations      - Optional. Number of additional times the hash function has been performed.
     name            - Optional. FQDN of the NSEC3PARAM record owner (zone name).
     salt            - Optional. Salt value.
     ttl             - Optional. TTL of the record.
     view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
     zone            - Optional. A zone name in FQDN format.
```

Example

```
#get the NSEC3PARAM record for the zone 'test.com'
my $nsec3param=$session->get(
                            object => "Infoblox::DNS::Record::NSEC3PARAM",
                            name   => "test.com"
```

```
                                );

#get all the NSEC3PARAM records for the default DNS view
my @retrieved_objs=$session->get(
                                    object => "Infoblox::DNS::Record::NSEC3PARAM",
                                    view   => "default"
                                );
```

Infoblox::Session->search( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to search the NSEC3PARAM Record object(s):

    flags           - Optional. Integer containing 8 one-bit flags that can be used to indicate
different processing.
    hash_algorithm  - Optional. Integer representing the cryptographic hash algorithm.
    iterations      - Optional. Number of additional times the hash function has been performed.
    name            - Optional. FQDN of the NSEC3PARAM record owner (zone name). Regular expression.
    salt            - Optional. Salt value (regular expression).
    ttl             - Optional. TTL of the record.
    view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
    zone            - Optional. A zone name in FQDN format.
```

Example

```
# search for all NSEC3PARAM records in the "domain.com" zone of the default DNS view
# Note that the 'default' DNS view is assumed implicitly here
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::NSEC3PARAM",
    zone   => "domain.com",
   );
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of an NSEC3PARAM Record object.

flags( )

Use this method to retrieve the flags field of an NSEC3PARAM Record object.

Parameter

none

Returns

The method returns the integer that contains 8 one-bit flags that can be used to indicate different processing. Currently only the least significant bit of the flags field is meaningful. It represents the opt-out flag, which is used to determine the processing of non-secure delegations.

Example

```
#Get the flags field.
my $flags = $nsec3param->flags();
```

hash_algorithm( )

Use this method to retrieve the cryptographic hash algorithm used to construct the hash value of the corresponding NSEC3 Record objects.

Parameter

none

Returns

The method returns an integer that corresponds to the cryptographic hash algorithm that is used to construct the hash value.

Example

```
#Get hash algorithm
my $hash_algorithm = $nsec3param->hash_algorithm();
```

iterations( )

Use this method to retrieve the number of iterations of the hash function performed to obtain the hashed name.

Parameter

none

Returns

The method returns the number of additional times the hash function has been performed to compute the hash value. This is an integer in the range 0 through 65535, inclusive.

Example

```
#Get the iterations parameter
my $iterations = $nsec3param->iterations();
```

name( )

Use this method to retrieve the owner name of the NSEC3PARAM record in FQDN format. The name value of the NSEC3PARAM record is the same as the name of the zone that contains this record.

Parameter

none

Returns

The method returns the name of the NSEC3PARAM record. This name is a zone name in FQDN format.

Example

```
#Get the NSEC3PARAM name field
my $name = $nsec3param->name();
```

salt( )

Use this method to retrieve the salt value of an NSEC3PARAM record. The salt is used to calculate the hashed name of NSEC3 resource records to defend against pre-calculated dictionary attacks.

Parameter

none

Returns

The method returns a salt value. The returned value is a string containing a sequence of hexadecimal digits representing the salt or '-' if the salt is not used.

Example

```
#Get the salt
my $salt = $nsec3param->salt();
```

ttl( )

Use this method to retrieve the Time to Live (TTL) value of an NSEC3PARAM Record object.

Parameter

none

Returns

The method returns the TTL attribute value. The returned parameter is a 32-bit integer (range from 0 to 4294967295) that represents the duration, in seconds, that the record is cached. Zero indicates that the record should not be cached.

Example

```
#Get TTL
my $ttl = $nsec3param->ttl();
```

view( )

Use this method to retrieve the DNS view object that contains the NSEC3PARAM Record object.

Parameter

none

Returns

The method returns the Infoblox::DNS::View object that contains the NSEC3PARAM record.

Example

```
#Get the view
my $view = $nsec3param->view();
```

zone( )

Use this method to retrieve the zone name of an NSEC3PARAM record.

Parameter

none

Returns

Returns the name of the zone that contains the given NSEC3PARAM record.

Example

```
# Get zone
my $zone = $nsec3param->zone();
```

SAMPLE CODE

The following sample code demonstrates the session methods on an NSEC3PARAM Record object.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
       my $session = Infoblox::Session->new(
           master   => $host_ip,
           username => "admin",
           password => "infoblox"
       );
       unless ($session) {
          die("Construct session failed: ",
              $session->status_code() . ":" . $session->status_detail());
       }
       print "Session created successfully\n";

       #Enable DNSSEC in the default view

       my $default_view=$session->get(
                                   object=> "Infoblox::DNS::View",
                                   name => "default"
                                  );
       unless($default_view) {
           die("Getting the default view failed:",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
       print "Got the default view successfully\n";

       $default_view->dnssec_enabled("true")
           or die("Changing the dnssec_enabled in the default view failed:",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
       $session->modify($default_view)
           or die("Changing the dnssec_enabled in the default view failed:",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
```

#Create a DNSSEC signed zone

```
       print "Creating Member primary server for the zone\n";
       my $primary=Infoblox::DNS::Member->new(
                                            ipv4addr => $node_ip,
                                            name => $host_name
                                           );
```

```
unless($primary) {
   die("Unable to create primary server object: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

my $zone = Infoblox::DNS::Zone->new(
                                    name => "domain.com",
                                    primary => $primary
                                  );
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";

#Retrieving zone back from the server in order to sign it
$zone = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless($zone) {
   die("Retrieving zone back failed: ",
        Infoblox::status_code( ). ":". Infoblox::status_detail( ));
}
print "Zone retrieved for signing successfully.\n";

$zone->dnssec_ksk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_zsk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_ksk_size(640) &&
$zone->dnssec_zsk_size(640)
  or die("Changing the zone DNSSEC setting failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->modify($zone)
  or die("Modifying dnssec values in zone failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Zone modified successfully\n";

#Signing the zone
$zone->dnssec_signed("true")
   or die("Signing of the zone failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
       print "Zone signed successfully\n";
```

#Getting the NSEC3PARAM record by zone name

```
my $nsec3param=$session->get(
                             object => "Infoblox::DNS::Record::NSEC3PARAM",
                             name   => "domain.com",
                             view   => "default"
                            );
     unless($nsec3param) {
  die("Getting NSEC3PARAM record failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
     }
     print "Got NSEC3PARAM record successfully.\n";
```

#Searching for NSEC3PARAM objects using regular expressions

```
my @retrieved_objs=$session->search(
                                    object => "Infoblox::DNS::Record::NSEC3PARAM",
                                    name   => ".*com",
                                    salt   => ".*",
                                    iterations => 100,
                                    view   => "default"
                                  );
     unless(@retrieved_objs>0) {
  die("Searching for NSEC3PARAM objects failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
     }
```

```
          print "Search for the NSEC3PARAM objects successful, ".scalar(@retrieved_objs)." objects
found\n";

#Removing the created zone and cleaning up the view

      $session->remove($zone)
        or die("Unable to remove the zone: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
      print "Zone removed successfully\n";

      $default_view->dnssec_enabled("false")
      && $default_view->override_dnssec("false")
      && $session->modify($default_view)
       or die("Restoring dnssec_enabled value in the default view failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::DNS::Record::NSEC3, Infoblox::DNS::View, Infoblox::DNS::Zone

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::PTR - DNS PTR record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - discovered_name( )
  - discoverer( )
  - first_discovered( )
  - extensible_attributes( )
  - ipv4addr( )
  - ipv6addr( )
  - last_discovered( )
  - mac_address( )
  - name( )
  - netbios( )
  - network_component_description( )
  - network_component_ip( )
  - network_component_name( )
  - network_component_port_description( )
  - network_component_port_name( )
  - network_component_port_number( )
  - network_component_type( )
  - os( )
  - port_duplex( )
  - port_link_status( )
  - port_speed( )
  - port_status( )
  - port_vlan_description( )
  - port_vlan_name( )
  - port_vlan_number( )
  - ptrdname( )
  - ttl( )
  - views( )
  - v_cluster( )
  - v_datacenter( )
  - v_host( )
  - v_name( )
  - v_netadapter( )
  - v_switch( )
  - v_type( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::PTR - DNS PTR record object.

DESCRIPTION

A PTR (pointer) record maps an address to a domain name, and can be only added to a reverse mapping zone. To define a specific address-to-name mapping, add a PTR record to a previously defined authoritative reverse-mapping zone.To create a PTR record, you need to specify a domain name and host name.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
#Constructor for an IPv4 PTR object
my $bind_ptr = Infoblox::DNS::Record::PTR->new (
        ipv4addr => $ipv4addr,              #Required
        ptrdname => $fqdn,                  #Required
        comment  => $string,                #Optional / Default is empty
        disable  => "true" | "false"        #Optional / Default "false"
        extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
        last_discovered  => $string,        #Optional / Default is empty
        mac_address      => $mac_addr,      #Optional / Default is empty
        name     => $fqdn,                  #Optional
        ttl      => $num | undef,           #Optional / Default is undefined
        views    => [$View1],               #Optional / Default is "default" view
    );

#Constructor for an IPv6 PTR object
my $bind_ptr = Infoblox::DNS::Record::PTR->new (
        ipv6addr => $ipv6addr,              #Required
        ptrdname => $fqdn,                  #Required
        comment  => $string,                #Optional / Default is empty
        disable  => "true" | "false"        #Optional / Default "false"
        extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
        name     => $fqdn,                  #Optional
        ttl      => $nun | undef,           #Optional / Default is undefined
        views    => [$View1],               #Optional / Default is "default" view
    );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a DNS PTR record object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Constructor for an IPv4 PTR object
my $bind_ptr = Infoblox::DNS::Record::PTR->new (
    ipv4addr => "10.0.0.1",
    ptrdname => "bind_ptr.domain.com",
    comment => "this is a demo bind_ptr record",
    disable => "true",
    ttl => "33",
    mac_address => "1:1:1:1:1:1",
    views => [$view1]  );
 # Submit for addition
 my $response = $session->add( $bind_ptr );

#Constructor for an IPv6 PTR object
my $bind_ptr = Infoblox::DNS::Record::PTR->new(
    ptrdname     => "ip6host.this.com",
    comment  => " this is a demo ipv6 bind_ptr record ",
    disable => "true",
    ttl => "333",
    ipv6addr => "1234:5678:9abc:1000::1");
 # Submit for addition
 my $response = $session->add( $bind_ptr );
```

Infoblox::Session->get( )

Use this method to retrieve existing objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get a specific DNS PTR object:

 name | ipv4addr | ipv6addr | ptrdname - Optional. name/ptrdname is in FQDN format. ipv4addr is an
IPv4 address with a 32-bit number in dotted decimal notation. ipv6addr is the IPv6 (128 bits)
address of the DNS PTR record. ptrdname is the domain name of the DNS PTR record.
 view - Optional. The DNS view in which the PTR record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
 zone - Optional. A zone name in FQDN format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 v_cluster - Optional. Discovered VMware cluster name.
```

```
v_datacenter - Optional. Discovered VMware datacenter name.
v_host     - Optional. Discovered VMware server name.
v_name     - Optional. Discovered VMware host or virtual machine name.
v_netadapter - Optional. Discovered VMware physical network adapter name.
v_switch   - Optional. Discovered VMware virtual switch name.
v_type     - Optional. Discovered VMware entity type (possible values: "VirtualMachine",
"HostSystem", "VirtualCenter").
```

Example

```
#To get an IPv4 PTR object by name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    name   => "1.0.0.10.in-addr.arpa",
    view   => "default" );

#To get an IPv4 PTR object by ipv4addr
my @retrieved_objs = $session->get(
    object   => "Infoblox::DNS::Record::PTR",
    ipv4addr => "10.0.0.1",
    view     => "default" );

#To get an IPv4 PTR object by ptrdname
my @retrieved_objs = $session->get(
    object   => "Infoblox::DNS::Record::PTR",
    ptrdname => "bind_ptr.domain.com",
    view     => "default" );

#To get an IPv6 PTR object by name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    name   => "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.c.b.a.9.8.7.6.5.4.3.2.1.ip6.arpa",
    view   => "default" );

#To get an IPv6 PTR object by ipv6addr
my @retrieved_objs = $session->get(
    object   => "Infoblox::DNS::Record::PTR",
    ipv6addr => "1234:5678:9abc:1000::1",
    view     => "default" );

#To get an IPv6 PTR object by ptrdname
my @retrieved_objs = $session->get(
    object   => "Infoblox::DNS::Record::PTR",
    ptrdname => "ip6host.this.com",
    view     => "default" );

# get all PTR object in zone "10.10.10.in-addr.arpa" of default view
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    zone   => "10.10.10.in-addr.arpa",
    view   => "default"
    );

# get all PTR objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    extensible_attributes => { 'Site' => 'Santa Clara' });

# get all PTR objects with a given v_type:
my @retrieved_objs = $session->get(
  object    => "Infoblox::DNS::Record::PTR",
  v_type    => "VirtualMachine");
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment for IPv4 or IPv6 object.
$bind_ptr->comment("this is a modified comment for bind PTR");
# Submit modification
my $response = $session->modify( $bind_ptr );
```

Infoblox::Session->remove( )

Use this function to submit to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

#Remove an object with a unique name

```
# Get the object with the same name for IPv4 object
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    name   => "1.0.0.10.in-addr.arpa",
    view   => "default" );
# find the desired object from retrieved list.
my $desired_ptr = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_ptr );

# Get the object with the same name for an IPv6 object
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    name   => "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.c.b.a.9.8.7.6.5.4.3.2.1.ip6.arpa",
    view   => "default" );
# find the desired object from retrieved list.
my $desired_ptr = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_ptr );
```

Infoblox::Session->search()

Use this method to search for existing DNS PTR record object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to search for a DNS PTR record object:

name | ipv4addr | ipv6addr | ptrdname    - Optional. name/ptrdname in FQDN (Fully Qualified Domain
Name) format (regular expression). ipv4addr is an IPv4 address with a 32-bit number in dotted
decimal notation. ipv6addr is the IPv6 address (128 bits) of the DNS PTR record. ptrdname is the
domain name of the DNS PTR record.
view    - Optional. The DNS view in which the PTR record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
comment - Optional. A comment in string format (regular expression).
extensible_attributes - Optional. A hash reference containing extensible attributes.
v_cluster - Optional. Discovered VMware cluster name (regular expression).
v_datacenter - Optional. Discovered VMware datacenter name (regular expression).
v_host  - Optional. Discovered VMware server name (regular expression).
v_name  - Optional. Discovered VMware host or virtual machine name (regular expression).
v_netadapter - Optional. Discovered VMware physical network adapter name (regular expression).
v_switch - Optional. Discovered VMware virtual switch name (regular expression).
v_type  - Optional. Discovered VMware entity type (possible values: "VirtualMachine", "HostSystem",
"VirtualCenter").
```

Example

```
# search for all DNS IPv4 PTR objects in the default DNS view that match 10.0.0.0/8 and that contain
"bind" in the comment field
my @retrieved_objs = $session->search(
    object  => "Infoblox::DNS::Record::PTR",
    name    => "10.in-addr.arpa",
    comment => "bind*",
    view    => "default" );

my @retrieved_objs = $session->search(
    object   => "Infoblox::DNS::Record::PTR",
    ipv4addr => '10\..*\..*\..*',
    comment  => "bind*",
    view     => "default" );

my @retrieved_objs = $session->search(
    object   => "Infoblox::DNS::Record::PTR",
    ptrdname => "bind_ptr.domain.com",
    comment  => "bind*",
    view     => "default" );

# search for all DNS IPv6 PTR objects under 1234:5678:9abc::/48 which have bind in comment field in
default view
my @retrieved_objs = $session->search(
    object  => "Infoblox::DNS::Record::PTR",
    name    => "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.c.b.a.9.8.7.6.5.4.3.2.1.ip6.arpa",
    comment => "bind*",
    view    => "default" );
```

```
my @retrieved_objs = $session->search(
    object   => "Infoblox::DNS::Record::PTR",
    ipv6addr => "1234:5678:9abc:1000::1",
    comment  => "bind*",
    view     => "default" );

my @retrieved_objs = $session->search(
    object   => "Infoblox::DNS::Record::PTR",
    ptrdname => "ip6host.this.com",
    comment  => "bind*",
    view     => "default" );

# search for all PTR objects in the "10.10.10.in-addr.arpa" zone of the default view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::PTR",
    zone   => "10.10.10.in-addr.arpa",
    view   => "default"
    );

# search all PTR records with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::Record::MX",
   extensible_attributes => { 'Site' => 'Santa Clara' });

# search all PTR records by v_datacenter:
my @retrieved_objs = $session->search(
   object       => "Infoblox::DNS::Record::PTR",
   v_datacenter => "test.*");
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute value of a PTR record.

comment( )

Use this method to set or retrieve the comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get comment
my $comment = $bindptr->comment();
#Modify comment
$bindptr->comment("Modifying the DNS PTR comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS record.

Include the specific parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false, therefore the DNS record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get disable
my $disable = $bindptr->disable();
#Modify disable
$bindptr->disable("true");
```

discovered_name( )

Use this method to retrieve the discovered name of a PTR record object.

Parameter

none

Returns

The method returns the discovered name.

Example

```
#Get discovered_name
my $discovered_name = $bindptr->discovered_name();
```

discoverer( )

Use this method to retrieve the discoverer of a PTR record object.

Parameter

none

Returns

The method returns the discoverer value.

Example

```
#Get discoverer
my $discoverer = $bindptr->discoverer();
```

first_discovered( )

Use this method to retrieve the date and time that the PTR record object was first discovered.

Parameter

none

Returns

The method returns the date and time that the PTR record object was first discovered.

Example

```
#Get first_discovered
my $first_discovered = $bindptr->first_discovered();
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a PTR record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $bindptr->extensible_attributes();
#Modify extensible attributes
$bindptr->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

ipv4addr( )

Use this method to set or retrieve the IPv4 address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get ipv4addr
my $ipv4addr = $bindptr->ipv4addr();
#Modify ipv4addr
$bind_ptr->ipv4addr("10.0.0.2");
```

ipv6addr( )

Use this method to set or retrieve the IPv6 address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The IPv6 (128 bits) of the DNS PTR record (example: aaaa:bbbb:cccc:dddd:eeee:ffff:0000:1111).

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get ipv6addr
my $ipv6addr = $bindptr->ipv6addr();
#Modify ipv6addr
$bind_ptr->ipv6addr("1234:5678:9abc:1000::2");
```

last_discovered( )

Use this method to retrieve the time this object was last seen by a discovery job.

The default value for this field is empty.

Parameter

N/A

Returns

The method returns the attribute value.

Example

```
#Get last_discovered
my $last_discovered = $bind_ptr->last_discovered();
```

mac_address( )

Use this method to set or retrieve the the mac address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

MAC Address of the PTR record object. Format for the hexadecimal strings for MAC addresses and vendor prefixes with colons or dashes. Both of the following formats are acceptable: 11:11:11:11:11:11 and 11-11-11-11-11-11.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mac_address
my $mac_address = $bind_ptr->mac_address();
#Modify mac_address
$bind_ptr->mac_address("11::11::11::11::11::12");
```

name( )

Use this method to retrieve the name. If modifying, use ipv4addr or ipv6addr methods.

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get name
my $name = $bindptr->name();
```

netbios( )

Parameter

None

Use this method to retrieve the name in the NetBIOS reply that responded to a NetBIOS query.

Returns

The method returns the attribute value.

Example

```
#Get netbios
my $netbios = $bind_ptr->netbios();
```

network_component_description( )

Use this method to retrieve the description of the network device that is connected to the PTR record object.

Parameter

none

Returns

The method returns the description of the network device.

Example

```
#Get network_component_description
my $network_component_description = $bindptr->network_component_description();
```

network_component_ip( )

Use this method to retrieve the IP address of the network device that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device IP address.

Example

```
#Get network_component_ip
my $network_component_ip = $bindptr->network_component_ip();
```

network_component_name( )

Use this method to retrieve the name of the network device that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device name.

Example

```
#Get network_component_name
my $network_component_name = $bindptr->network_component_name();
```

network_component_port_description( )

Use this method to retrieve the description of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the description of the network device port.

Example

```
#Get network_component_port_description
my $network_component_port_description = $bindptr->network_component_port_description();
```

network_component_port_name( )

Use this method to retrieve the name of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device port name.

Example

```
#Get network_component_port_name
my $network_component_port_name = $bindptr->network_component_port_name();
```

network_component_port_number( )

Use this method to retrieve the number of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device port number value in the range 0-9999.

Example

```
#Get network_component_port_number
my $network_component_port_number = $bindptr->network_component_port_number();
```

network_component_type( )

Use this method to retrieve the type of the network device that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device type.

Example

```
#Get network_component_type
my $network_component_type = $bindptr->network_component_type();
```

os( )

Use this method to retrieve the operating system associated with this PTR record.

Returns

The method returns the attribute value.

Example

```
#Get os
my $os = $bind_ptr->os();
#Modify os
$bind_ptr->os("os-name");
```

port_duplex( )

Use this method to retrieve the duplex setting of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device port duplex setting. Returned values, if any, are one of the following:

```
Full
Half
```

Example

```
#Get port_duplex
my $port_duplex = $bindptr->port_duplex();
```

port_link_status( )

Use this method to retrieve the link status of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device port link status. Returned values, if any, are one of the following:

```
Connected
Not Connected
Unknown
```

Example

```
#Get port_link_status
my $port_link_status = $bindptr->port_link_status();
```

port_speed( )

Use this method to retrieve the speed of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device port speed value. Returned values, if any, are one of the following:

```
10M
100M
1G
10G
100G
Unknown
```

Example

```
#Get port_speed
my $port_speed = $bindptr->port_speed();
```

port_status( )

Use this method to retrieve the status of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the network device port status. Returned values, if any, are one of the following:

```
Up
Down
Unknown
```

Example

```
#Get port_status
my $port_status = $bindptr->port_status();
```

port_vlan_description( )

Use this method to retrieve the VLAN description of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the VLAN description of the network device port that is connected to the PTR record object.

Example

```
#Get port_vlan_description
my $port_vlan_description = $bindptr->port_vlan_description();
```

port_vlan_name( )

Use this method to retrieve the VLAN name of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the VLAN name of the network device port.

Example

```
#Get port_vlan_name
my $port_vlan_name = $bindptr->port_vlan_name();
```

port_vlan_number( )

Use this method to retrieve the VLAN number of the network device port that is connected to the PTR record object.

Parameter

none

Returns

The method returns the port VLAN number in the range 0-9999.

Example

```
#Get port_vlan_number
my $port_vlan_number = $bindptr->port_vlan_number();
```

ptrdname( )

Use this method to set or retrieve the PTR domain name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the domain name for the DNS PTR record.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get ptrdname
my $ptrdname = $bindptr->ptrdname();
#Modify ptrdname
$bindptr->ptrdname("new domain name");
```

ttl( )

Use this method to configure or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined; therefore the record inherits the TTL value from the zone level.

With a specified TTL value, the method overrides the zone values with the specified TTL value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration ,in seconds, that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get ttl
my $ttl = $bindptr->ttl();
#Modify ttl
$bindptr->ttl(1800);
#Un-override ttl
$bindptr->ttl(undef);
```

views( )

Use this method to specify or retrieve the view in which the PTR record is located.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the default view therefore the PTR record is referenced under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get views
my $ref_views = $bindptr->views();
#Modify views, list of Infoblox::DNS::View objects
$bindptr->views([$view1]);
```

v_cluster( )

Use this method to retrieve the name of the VMware cluster associated with the PTR Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware cluster name. The default value is an empty string.

Example

```
#Get v_cluster
my $v_cluster = $bindptr->v_cluster();
```

v_datacenter( )

Use this method to retrieve the name of the VMware datacenter associated with the PTR Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware datacenter name. The default value is an empty string.

Example

```
#Get v_datacenter
my $v_datacenter = $bindptr->v_datacenter();
```

v_host( )

Use this method to retrieve the name of the VMware host associated with the PTR Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware host name. The default value is an empty string.

Example

```
#Get v_host
my $v_host = $bindptr->v_host();
```

v_name( )

Use this method to retrieve the name of the VMware entity associated with the PTR Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware entity name. The default value is an empty string.

Example

```
#Get v_name
my $v_name = $bindptr->v_name();
```

v_netadapter( )

Use this method to retrieve the name of the physical network adapter through which the VMware entity is connected to the appliance. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the network adapter name. The default value is an empty string.

Example

```
#Get v_netadapter
my $v_netadapter = $bindptr->v_netadapter();
```

v_switch( )

Use this method to retrieve the name of the virtual switch through which the VMware entity is connected to the appliance. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the virtual switch name. The default value is an empty string.

Example

```
#Get v_switch
my $v_switch = $bindptr->v_switch();
```

v_type( )

Use this method to retrieve the type of VMware entity associated with the PTR Record object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware entity name. The default value is an empty string. The returend value, if any, can be one of the following:

```
VirtualMachine
HostSystem
VirtualCenter
```

Example

```
#Get v_type
my $v_type = $bindptr->v_type();
```

zone( )

Use this method to retrieve the zone name of a DNS PTR record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $bindptr->zone();
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object, such as create, search, modify, and remove an object. This sample code also includes error handling for the operations.

#Preparation prior to a DNS IPv4 PTR record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
```

```
}
print "Session created successfully\n";

#Create the zone prior to a PTR record insertion
my $zone = Infoblox::DNS::Zone->new(name => "10.0.0.0/8");
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "10.in-addr.arpa");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS IPv4 PTR record

```
my $bind_ptr = Infoblox::DNS::Record::PTR->new(
   ptrdname     => "bind_ptr.domain.com",
   comment  => " this is a demo bind_ptr record ",
   ipv4addr => "10.0.0.1"
 );
 unless ($bind_ptr) {
    die("Construct DNS record PTR failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
 }
print "DNS PTR object created successfully\n";

#Add the DNS PTR record object to the Infoblox appliance through a session
$session->add($bind_ptr)
    or die("Add record PTR failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS PTR object added to server successfully\n";
```

#Search for a specific DNS IPv4 PTR record

```
#Search for all PTR records that match "10.in-addr.arpa"
my @retrieved_objs = $session->search(
    "object" => "Infoblox::DNS::Record::PTR",
    "name" => '10\.in-addr.arpa',
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record PTR failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS PTR object found at least 1 matching entry\n";

#Search all PTR records that have "bind" in the comment
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::PTR",
    name   => '10\.*\.in-addr.arpa',
    comment => "bind*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record PTR failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS PTR object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS IPv4 PTR record

```
#Get a PTR record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    name   => "1.0.0.10.in-addr.arpa",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die
```

```
        ("Get record PTR failed: ",
         $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS PTR object found at least 1 matching entry\n";
#Modify one of the attributes of the specified PTR record
$object->comment ("modified comment");

#Apply the changes
$session->modify($object)
    or die("Modify record PTR failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS PTR object modified successfully \n";
```

#Remove a DNS IPv4 PTR record

```
#Get a PTR record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    name   => "1.0.0.10.in-addr.arpa",
);

 my $object = $retrieved_objs[0];
  unless ($object) {
    die("Get record PTR failed: ",
         $session->status_code() . ":" . $session->status_detail());
  }
  print "Get DNS PTR object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove record PTR failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS PTR object removed successfully \n";
```

#Preparation prior to a DNS IPv6 PTR record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";

#Create the zone prior to a PTR record insertion
my $zone = Infoblox::DNS::Zone->new(name => "1234:5678:9abc::/48");
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "c.b.a.9.8.7.6.5.4.3.2.1.ip6.arpa");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS IPv6 PTR record

```
my $bind_ptr = Infoblox::DNS::Record::PTR->new(
   ptrdname     => "ip6host.this.com",
   comment  => " this is a demo bind_ptr record ",
   ipv6addr => "1234:5678:9abc:1000::1"
 );
```

```
 unless ($bind_ptr) {
    die("Construct DNS record PTR failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
 }
print "DNS PTR object created successfully\n";
#Add the DNS PTR record object to the Infoblox  appliance through a session
       $session->add($bind_ptr)
           or die("Add record PTR failed: ",
                  $session->status_code() . ":" . $session->status_detail());
       print "DNS PTR object added to server successfully\n";
```

#Search for a specific DNS IPv6 PTR record

```
#Search for a PTR record in the zone
my @retrieved_objs = $session->search(
    "object" => "Infoblox::DNS::Record::PTR",
    "name" => "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.c.b.a.9.8.7.6.5.4.3.2.1.ip6.arpa",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record PTR failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS PTR object found at least 1 matching entry\n";
```

#Get and modify a DNS IPv6 PTR record

```
#Get PTR record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    name   => "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.c.b.a.9.8.7.6.5.4.3.2.1.ip6.arpa",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record PTR failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS PTR object found at least 1 matching entry\n";

#Modify one of the attributes of the specified PTR record
$object->comment ("modified comment");

#Apply the changes
$session->modify($object)
    or die("Modify record PTR failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DNS PTR object modified successfully \n";
```

#Remove a DNS IPv6 PTR record

```
#Get PTR record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::PTR",
    name   => "1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.c.b.a.9.8.7.6.5.4.3.2.1.ip6.arpa",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record PTR failed: ",
         $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS PTR object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove record PTR failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS PTR object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::RRSIG - RRSIG Record object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
- METHODS
  - algorithm( )
  - expiration( )
  - inception( )
  - key_tag( )
  - labels( )
  - name( )
  - original_ttl( )
  - signature( )
  - signer_name( )
  - ttl( )
  - type_covered( )
  - view( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::RRSIG - RRSIG Record object

DESCRIPTION

RRSIG records are one of the resource records in DNSSEC. These records store digital signatures of resource record sets (RRsets). The digital signatures are used to authenticate the data that is in the signed RRsets.

RRSIG resource records are defined in RFC 4034.

RRSIG records are automatically generated upon the signing of an authoritative zone. The RRSIG record-object is read-only.

CONSTRUCTOR

The Infoblox::DNS::Record::RRSIG object is a read-only object and does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to an RRSIG record object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
  Apply the following attributes to get the RRSIG Record object(s):

     algorithm       - Optional. The public key encryption algorithm. The allowed values are
'1','3','5','6','7','RSAMD5','RSASHA1','DSA','NSEC3RSASHA1','NSEC3DSA','RSASHA256','RSASHA512','NSEC3RS
and 'NSEC3RSASHA512'.
     expiration      - Optional. Signature expiration time, in seconds, since 1 January 1970 00:00:00
UTC.
     name            - Optional. The name of the corresponding RRset.
     inception       - Optional. Signature inception time, in seconds, since 1 January 1970 00:00:00
UTC.
     key_tag         - Optional. Key tag identifying the corresponding public key in the DNSKEY
record.
     labels          - Optional. Specifies the number of labels in the original RRSIG RR owner name.
     original_ttl    - Optional. TTL of the covered RRset as it appears in the authoritative zone.
     signature       - Optional. Cryptographic signature that covers the RRSIG RDATA.
     signer_name     - Optional. Domain name of the zone that contains the signed RRset.
     ttl             - Optional. TTL value of the record itself.
```

```
     type_covered    - Optional. Identifies the type of the RRset that is covered by this RRSIG
record.
     view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
     zone            - Optional. A zone name in FQDN format.
```

Example

```
# get all the RRSIG records for the zone 'domain.com' in the default DNS view.
my @retrieved_objs = $session->get(
                     object => "Infoblox::DNS::Record::RRSIG",
                     signer_name   => "domain.com",
                     view   => "default" );

# get the RRSIG record that corresponds to the 'test.domain.com' A record of the default DNS view
my $retrieved_obj = $session->get(
                     object    => "Infoblox::DNS::Record::RRSIG",
                     name      => "test.domain.com",
                     type_covered => "A",
                     view      => "default" );
```

Infoblox::Session->search( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
 Apply the following attributes to search the RRSIG Record object(s):

     algorithm       - Optional. The public key encryption algorithm. The allowed values are
'1','3','5','6','7','RSAMD5','RSASHA1','DSA','NSEC3RSASHA1','NSEC3DSA','RSASHA256','RSASHA512','NSEC3RS
and'NSEC3RSASHA512'.
     expiration      - Optional. Signature expiration time, in seconds, since 1 January 1970 00:00:00
UTC. Searches for the exact value.
     expiration_start- Optional. Used together with the expiration_end attribute to search for the
expiration time in a particular time interval.
                                 Marks the start of the search interval.
     expiration_end  - Optional. Marks the end of the expiration time interval in the search
interval.
     name            - Optional. The name of the corresponding RRset (regular expression).
     inception       - Optional. Signature inception time, in seconds, since 1 January 1970 00:00:00
UTC. Searches for the exact value.
     inception_start - Optional. Used together with the inception_end attribute to search for the
inception time in a particular time interval.
                                 Marks the start of the search interval.
     inception_end   - Optional. Marks the end of the inception time interval in the search interval.
     key_tag         - Optional. Key tag that identifies the corresponding public key in the DNSKEY
record.
     labels          - Optional. Specifies the number of labels in the original RRSIG RR owner name.
     original_ttl    - Optional. TTL of the covered RRset as it appears in the authoritative zone.
     signature       - Optional. Cryptographic signature that covers the RRSIG RDATA (regular
expression).
     signer_name     - Optional. Domain name of the zone that contains the signed RRset (regular
expression).
     ttl             - Optional. TTL value of the record itself.
     type_covered    - Optional. Identifies the type of the RRset that is covered by this RRSIG
record (regular expression).
     view            - Optional. The DNS view in which the record is located. By default, all DNS
views are searched. But if you omit this attribute and specify a zone, the appliance searches the
'default' view only.
     zone            - Optional. A zone name in FQDN format.
```

Example

```
# get all the RRSIG records for the zone matching the regular expression.
my @retrieved_objs = $session->search(
                     object => "Infoblox::DNS::Record::RRSIG",
                     signer_name   => ".*[.]com",);

# get all the RRSIG records with inception time between 12:00 September 1, 2009 and 12:00 September
10, 2009, expiring no later then January 1, 2010.
my @retrieved_objs = $session->search(
                     object            => "Infoblox::DNS::Record::RRSIG",
                     inception_start   => 1254398400,
                     inception_end     => 1255176000,
                     expiration_end    => 1264982400);

# search for all RRSIG records in the "domain.com" zone of the default DNS view
# Note that the 'default' DNS view is assumed implicitly here
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::RRSIG",
    zone   => "domain.com",
   );
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of an RRSIG Record object.

algorithm( )

Use this method to retrieve the value of the public key encryption algorithm used to create the RRSIG signature.

Parameter

none

Returns

The method returns the public key encryption algorithm. Returned values are one of the following:

RSAMD5
DSA
RSASHA1
NSEC3DSA
NSEC3RSASHA1
RSASHA256
RSASHA512

Example

```
#Get the public key encryption algorithm.
my $algorithm = $rrsig->algorithm();
```

expiration( )

Use this method to retrieve the expiration time of an RRSIG record.

Parameter

none

Returns

The method returns the RRSIG record expiration time. The returned value is in UNIX timestamp format; that is, it is measured in seconds since 00:00:00 January 1, 1970 UTC.

Example

```
#Get the RRSIG expiration time
my $expiration = $rrsig->expiration();
```

inception( )

Use this method to retrieve the RRSIG inception time. This is the time when the record was created.

Parameter

none

Returns

The method returns the RRSIG record inception time. The returned value is in UNIX timestamp format; that is, it is measured in seconds since 00:00:00 January 1, 1970 UTC.

Example

```
#Get the inception time of the RRSIG record
my $inception = $rrsig->inception();
```

key_tag( )

Use this method to retrieve the key tag of the public key that correspods to the RRSIG Record object.

Parameter

none

Returns

The method returns the key tag of the public key that corresponds to the private key used to encrypt RRSIG data. The key tag is

a 16-bit integer key fingerprint that is used to distinguish one key from another when multiple keys are associated with the same zone.

Example

```
#Get the key tag of the public key
my $key_tag = $rrsig->key_tag();
```

labels( )

Use this method to retrieve the number of labels in the name of the RRset signed with the RRSIG object.

Parameter

none

Returns

The method returns the number of labels in the original RRSIG RR owner name. Usually this number is the same as the number of labels in the 'name' field.

Example

```
#Get the number of labels in the original RRSIG RR owner name.
my $labels = $rrsig->labels();
```

name( )

Use this method to retrieve the fully qualifyed domain name of the RRset covered by the RRSIG record.

Parameter

none

Returns

The method returns the name attribute of the RRSIG object. This is a name of the RRset covered by the RRSIG record.

Example

```
#Get the name attribute
my $fqdn = $rrsig->name();
```

original_ttl( )

Use this method to retrieve the TTL of the covered RRset as it appears in the authoritative zone of an RRSIG record object.

Parameter

none

Returns

The method returns the TTL of the covered RRset as it appears in the authoritative zone attribute value.

Example

```
#Get the TTL of the covered RRset
my $original_ttl = $rrsig->original_ttl();
```

signature( )

Use this method to retrieve the cryptographic signature that covers the RRSIG RDATA of an RRSIG Record object.

Parameter

none

Returns

The method returns the cryptographic signature that covers the RRSIG RDATA attribute value.

Example

```
#Get the signature that covers the RRSIG RDATA
my $signature = $rrsig->signature();
```

signer_name( )

Use this method to retrieve the domain name of the zone that contains the signed RRset.

Parameter

none

Returns

The method returns the domain name of the zone that contain the signed RRset in FQDN format.

Example

```
#Get the signer name
my $signer_name = $rrsig->signer_name();
```

ttl( )

Use this method to retrieve the Time to Live (TTL) value of an RRSIG Record object.

Parameter

none

Returns

The method returns the TTL attribute value. The returned parameter is a 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Example

```
#Get the TTL of the RRSIG record
my $ttl = $rrsig->ttl();
```

type_covered( )

Use this method to retrieve the type of the RRset that is covered by this RRSIG record.

Parameter

none

Returns

The method returns the type of the RRset that is covered by this RRSIG record attribute value.

Example

```
#Get the type of the RRset that is covered by this RRSIG record
my $type_covered = $rrsig->type_covered();
```

view( )

Use this method to retrieve the DNS view object that contains the RRSIG Record object.

Parameter

none

Returns

The method returns the Infoblox::DNS::View object that contains the RRSIG record.

Example

```
#Get the DNS view
my $view = $rrsig->view();
```

zone( )

Use this method to retrieve the zone name of an RRSIG record.

Parameter

none

Returns

Returns the zone name of the zone that contains RRSIG record. The name is in FQDN format.

Example

```
# Get zone
my $zone = $rrsig->zone();
```

SAMPLE CODE

The following sample code demonstrates the session methods on an RRSIG record object.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
       my $session = Infoblox::Session->new(
           master   => "192.168.1.2",
           username => "admin",
           password => "infoblox"
       );
       unless ($session) {
          die("Construct session failed: ",
              $session->status_code() . ":" . $session->status_detail());
       }
       print "Session created successfully\n";

       #Enable DNSSEC in the default view

       my $default_view=$session->get(
                                   object=> "Infoblox::DNS::View",
                                   name => "default"
                                  );
       unless($default_view) {
           die("Getting the default view failed:",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
       print "Got the default view successfully\n";

       $default_view->dnssec_enabled("true")
           or die("Changing the dnssec_enabled in the default view failed:",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
       $session->modify($default_view)
           or die("Changing the dnssec_enabled in the default view failed:",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
```

#Creating a signed zone and populationg it with an A record

```
       print "Creating Member primary server for the zone\n";
       my $primary=Infoblox::DNS::Member->new(
                                         ipv4addr => "192.168.1.2",
                                         name => "infoblox.localdomain"
                                        );
       unless($primary) {
          die("Unable to create primary server object: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
       }

       my $zone = Infoblox::DNS::Zone->new(
                                       name => "domain.com",
                                       primary => $primary
                                      );
       unless ($zone) {
          die("Construct zone failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
       print "Zone object created successfully\n";

       #Verify if the zone exists
       my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
       unless ($object) {
          print "Zone does not exist on server, safe to add the zone\n";
          $session->add($zone)
             or die("Add zone failed: ",
                    $session->status_code() . ":" . $session->status_detail());
```

```
}
print "Zone added successfully\n";

#Adding A record to the zone to demonstrate the corresponding RRSIG record
my $a_record=Infoblox::DNS::Record::A->new(
                                          ipv4addr => "10.9.8.7",
                                          name => "recorda.domain.com"
                                        );
unless($a_record) {
   die("Creating A record failed : ",
      Infoblox::status_code() . ":" . Infoblox::status_detail());
}

$object = $session->get( object => "Infoblox::DNS::Record::A", name => "recorda.domain.com" );
unless ($object) {
   print "A record does not exist on the server, safe to add the A record\n";
   $session->add($a_record)
     or die("Adding A record failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "A record added to the zone successfully\n";

#Retrieving zone back from the server in order to sign it
$zone = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless($zone) {
   die("Retrieving zone back failed: ",
        Infoblox::status_code( ). ":". Infoblox::status_detail( ));
}
print "Zone retrieved for signing successfully.\n";

$zone->dnssec_ksk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_zsk_algorithm("NSEC3RSASHA1") &&
$zone->dnssec_ksk_size(640) &&
$zone->dnssec_zsk_size(640)
  or die("Changing the zone DNSSEC setting failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->modify($zone)
  or die("Modifying dnssec values in zone failed: ",
         Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Zone modified successfully\n";

#Signing the zone
$zone->dnssec_signed("true")
   or die("Signing of the zone failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
       print "Zone signed successfully\n";
```

#Getting the RRSIG record corresponding to A record

```
my $rrsig_record_a=$session->get(
                            object => "Infoblox::DNS::Record::RRSIG",
                            name   => "recorda.domain.com",
                            signer_name => "domain.com",
                            type_covered   => "A",
                            view   => "default"
                           );
     unless($rrsig_record_a) {
  die("Getting RRSIG corresponding to the A record failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
     }
     print "Got RRSIG record successfully\n";
```

#Searching for RRSIG objects using regular expressions

```
my @retrieved_objs=$session->search(
                                   object => "Infoblox::DNS::Record::RRSIG",
                                   name   => ".*com",
                                   type_covered => "A|MX",
                                   view   => "default"
                                 );
     unless(@retrieved_objs>0) {
  die("Searching for RRSIG objects failed: ",
     Infoblox::status_code() . ":" . Infoblox::status_detail());
     }
     print "Search for the RRSIG objects successful, ".scalar(@retrieved_objs)." objects found\n";
```

```
#Searching for RRSIG objects created today using inception time interval search
@retrieved_objs=$session->search(
                                    object => "Infoblox::DNS::Record::RRSIG",
                                    name   => ".*com",
                                    inception_start => (time() -86400),
                                    inception_end => time(),
                                    view   => "default"
                                  );
     unless(@retrieved_objs>0) {
   die("Searching for RRSIG objects failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
     }
     print "Search for the RRSIG objects successful, ".scalar(@retrieved_objs)." objects found\n";
```

#Removing the created zone and cleaning up the view

```
$session->remove($zone)
  or die("Unable to remove the zone: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
print "Zone removed successfully\n";

$default_view->dnssec_enabled("false")
&& $default_view->override_dnssec("false")
&& $session->modify($default_view)
 or die("Restoring dnssec_enabled value in the default view failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::DNS::Record::DNSKEY, Infoblox::DNS::View, Infoblox::DNS::Zone

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::SRV - DNS SRV record object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - name( )
  - port( )
  - priority( )
  - target( )
  - ttl( )
  - views( )
  - weight( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::SRV - DNS SRV record object

DESCRIPTION

A DNS SRV object represents an SRV resource record, which is also known as a service record. An SRV record provides information on available services.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
 my $srv = Infoblox::DNS::Record::SRV->new(
     name     => $fqdn,                  #Required
     port     => $num,                   #Required
     priority => $num,                   #Required
     target   => $fqdn,                  #Required
     weight   => $num,                   #Required
     comment  => $string,                #Optional / Default is empty
     disable  => "true" | "false"        #Optional / Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl      => $num | undef,           #Optional / Default is undefined
     views    => [$View1],               #Optional / Default is "default" view
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a DNS SRV record object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct object
my $bind_srv = Infoblox::DNS::Record::SRV->new(
    name     => "bind_srv.domain.com",
    comment  => "add bind srv bind_srv.domain.com",
```

```
        priority => 1,
        port     => 1,
        weight   => 1,
        target   => "old.target.test.org",
    );
    # Submit for addition
    my $response = $session->add( $bind_srv );
```

Infoblox::Session->get( )

Use this method to retrieve the existing objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
    Apply the following attributes to get a specified DNS SRV object:

     name     - Optional. A name in string format.
     view     - Optional. The DNS view in which the SRV record is located. By default, all DNS views are
   searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
   view only.
     zone     - Optional. A zone name in FQDN format.
     weight   - Optional. The weight of the SRV record.
     port     - Optional. The port of the SRV record.
     priority - Optional. The priority of the SRV record.
     target   - Optional. The target of the SRV record.
     extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::SRV",
        name     => "bind_srv.domain.com",
        view   => "default" );

    # get all DNS SRV record in zone "domain.com" of default view
    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::SRV",
        zone     => "domain.com",
        view   => "default" );

    # get all DNS SRV records with a given extensible attribute
    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::SRV",
        extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
    # Use method to modify the comment.
    $bind_srv->comment("this is a modified comment");
    # Submit modification
    my $response = $session->modify( $bind_srv);
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
    # Get the object under the same name
    my @retrieved_objs = $session->get(
        object => "Infoblox::DNS::Record::SRV",
        name     => "bind_srv.domain.com",
        view   => "default" );
    my $desired_srv= $retrieved_objs[0];
    # Submit for removal
    my $response = $session->remove( $desired_srv );
```

Infoblox::Session->search( )

Use this method to search for existing DNS SRV record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to search for a DNS SRV record object:

   name    - Required. A name in FQDN (Fully Qualified Domain Name) format (regular expression).
   view    - Optional. The DNS view in which the SRV record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
   zone    - Optional. A zone name in FQDN format.
   comment - Optional . A comment in string format (regular expression).
   weight  - Optional. The weight of the SRV record (regular expression).
   port    - Optional. The port of the SRV record (regular expression).
   priority - Optional. The priority of the SRV record (regular expression).
   target  - Optional. The target of the SRV record (regular expression).
   extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS SRV objects in the default DNS view that match "domain.com" and that contain
"bind" in the comment field
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::SRV",
    name     => 'domain\.com',
    comment  => "bind",
    view     => "default" );

# search for all DNS SRV records in the "domain.com" zone of the default view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::SRV",
    zone     => "domain.com",
    view   => "default" );

# search all DNS SRV records with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::Record::SRV",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an SRV record.

comment( )

Use this method to add or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get comment
my $comment = $bindsrv->comment();
#Modify comment
$bindsrv->comment("Modifying the DNS SRV comment");
```

disable( )

Use this method to set or retrieve the disable flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false, therefore the DNS record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get disable
my $disable = $bindsrv->disable();
#Modify disable
$bindsrv->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS SRV record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $bindsrv->extensible_attributes();
#Modify extensible attributes
$bindsrv->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

name( )

Use this method to set or retrieve the name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the DNS SRV record.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get name
my $name = $bindsrv->name();
#Modify name
$bindsrv->name("modified_bind_srv.domain.com");
```

port( )

Use this method to add or retrieve a port.

Parameter

The port number for the DNS SRV record (range from 0 to 65535).

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get port
my $port = $bindsrv->port();
#Modify port
$bindport->port("2");
```

priority( )

Use this method to add or retrieve the priority.

Parameter

Number with the priority for the DNS SRV record (range from 0 to 65535).

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get priority
my $priority = $bindsrv->priority();
#Modify priority
$bindsrv->priority("2");
```

target( )

Parameter

Text with the target for the DNS SRV record.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get target
my $target = $bindsrv->target();
#Modify target
$bindsrv->target ("new.target.test.org");
```

ttl( )

Use this method to configure or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined; therefore the record inherits the TTL value from the zone level.

With a specified TTL value, the method overrides the zone values with the specified TTL value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get ttl
my $ttl = $bindsrv->ttl();
#Modify ttl
$bindsrv->ttl(1800);
#Un-override ttl
$bindsrv->ttl(undef);
```

views( )

Use this method to specify or retrieve the views in which the SRV record is located.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the default view, therefore the SRV record is located under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get views
```

```
my $ref_views = $bindsrv->views();
#Modify views, list of Infoblox::DNS::View objects
$bindsrv->views([$view1]);
```

weight( )

Use this method to set or retrieve the weight.

Parameter

The weight of the DNS SRV record.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get weight
my $weight = $bindsrv->weight ();
#Modify weight
$bindsrv->weight ("3");
```

zone( )

Use this method to retrieve the zone name of the DNS SRV record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $bindsrv->zone();
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object, such as create, search, modify, and remove an object. This sample code also includes error handling for the operations.

#Preparation prior to a DNS SRV record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";

#Create the zone prior to a SRV record insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
```

```
            $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS SRV record

```
my $bind_srv = Infoblox::DNS::Record::SRV->new(
   name     => "bind_srv.domain.com",
   comment  => "add bind bind_srv.domain.com",
   priority => 1,
   port     => 1,
   weight   => 1,
   target   => "old.target.test.org",
 );
 unless ($bind_srv) {
    die("Construct DNS record SRV failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
 }
print "DNS SRV object created successfully\n";

#Add the DNS SRV record object to Infoblox  appliance through a session
$session->add($bind_srv)
    or die("Add record SRV failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DNS SRV object added to server successfully\n";
```

#Search for a specific DNS SRV record

```
#Search for SRV records that match bind_srv.domain.com
my @retrieved_objs = $session->search(
    "object" => "Infoblox::DNS::Record::SRV",
    "name" => 'bind_srv\.domain\.com',
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record SRV failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS SRV object found at least 1 matching entry\n";

#Search all SRV records that match "domain.com" and that contain "bind" in the comment field
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::SRV",
    name   => 'domain\.com',
    comment => "bind"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record SRV failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS SRV object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS SRV record

```
#Get a SRV record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::SRV",
    name   => "bind_srv.domain.com",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record SRV failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS SRV object found at least 1 matching entry\n";

#Modify one of the attributes of the specified SRV record
$object->comment ("modified comment for SRV rec");

#Apply the changes
$session->modify($object)
    or die("Modify record SRV failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DNS SRV object modified successfully \n";
```

#Remove a DNS SRV record

```
#Get a SRV record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::SRV",
    name   => "bind_srv.domain.com",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record SRV failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS SRV object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove record SRV failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS SRV object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Record::TXT - DNS TXT record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - name( )
  - text( )
  - ttl( )
  - views( )
  - zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Record::TXT - DNS TXT record object.

DESCRIPTION

A TXT (text) record contains supplemental information for a host. SPF (Sender Policy Framework) records are specialized TXT records that identify the servers that send mail from a domain.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
 my $txt = Infoblox::DNS::Record::TXT->new(
     name      => $fqdn,                 #Required
     text      => $string                #Required
     comment   => $string,               #Optional / Default is empty
     disable   => "true" | "false"       #Optional / Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl       => $num | undef,          #Optional / Default is undefined
     views     => [$View1],              #Optional / Default is "default" view
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a TXT record object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the TXT object
my $bind_txt = Infoblox::DNS::Record::TXT->new(
    name    => "txt.domain.com",
    text    => "this is text string",
    comment => "add TXT txt.domain.com",
    disable => "false",
    ttl     => "360",
    views   => [$view1]  );
# Submit for addition
my $response = $session->add( $bind_txt );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS TXT object:

  name - Optional. Host name in string format.
  view - Optional. The DNS view in which the TXT record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  text - Optional. The text of the TXT record.
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::TXT",
    name   => "txt.domain.com",
    view   => "default" );

# get all DNS TXT records in the "domain.com" zone of the default view
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::TXT",
    zone   => "domain.com",
    view   => "default" );

# get all DNS TXT records with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::TXT",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$bind_txt->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $bind_txt );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::TXT",
    name   => "txt.domain.com",
    view   => "default" );
# find the desired object from the retrieved list.
my $desired_txt = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $ desired_txt );
```

Infoblox::Session->search( )

Use this method to search for DNS TXT record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS TXT record object:

  name    - Optional. A name in FQDN (Fully Qualified Domain Name) format (regular expression).
  view    - Optional. The DNS view in which the TXT record is located. By default, all DNS views are
searched. But if you omit this attribute and specify a zone, the appliance searches the 'default'
view only.
  zone    - Optional. A zone name in FQDN format.
  comment - Optional. A comment in string format (regular expression).
  text    - Optional. The text of the TXT record (regular expression).
```

extensible_attributes - Optional. A hash reference containing extensible attributes.

Example

```
# search for all DNS TXT objects that match "domain.com" in the default DNS view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::TXT",
    name   => 'domain\.com',
    view   => "default" );

# search for all DNS TXT records in the "domain.com" zone of the default view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::TXT",
    zone   => "domain.com",
    view   => "default" );

# search all DNS TXT records with a given extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::Record::TXT",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a TXT record

comment( )

Use this method to set or retrieve the descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $bind_txt->comment();
#Modify comment
$bind_txt->comment("Modifying the DNS TXT comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $bind_txt->disable();
#Modify disable
$bind_txt->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $bindtxt->extensible_attributes();
#Modify extensible attributes
$bindtxt->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

name( )

Use this method to set or retrieve the name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the DNS TXT object in Fully Qualified Domain Name (FQDN) format. The name can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $bind_txt->name();
#Modify name
$bind_txt->name("new.domain.com");
```

text( )

Use this method to set or retrieve the text.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text that you want to associate with the record. It can contain up to 255 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $bind_txt->text();
#Modify name
$bind_txt->text("this is new text string");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $bind_txt->ttl();
#Modify ttl
$bind_txt->ttl(1800);
#Un-override ttl
$bind_txt->ttl(undef);
```

views( )

Use this method to set or retrieve the view of a TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the "default" view, which means the TXT record is located under the default view.

Parameter

Array reference of defined Infoblox::DNS::View objects.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $bind_txt->views();
#Modify views, list of Infoblox::DNS::View objects
$bind_txt->views([$view1]);
```

zone( )

Use this method to retrieve the zone name of a DNS TXT record. This method is read-only and cannot be set.

Parameter

None

Returns

Returns the attribute value.

Example

```
# Get zone
my $zone = $bind_txt->zone();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS TXT record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
```

```
my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
               );

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the zone prior to a TXT record insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
       die("Construct zone failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
       print "Zone does not exist on server, safe to add the zone\n";
       $session->add($zone)
               or die("Add zone failed: ",
                               $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS TXT record

```
my $bind_txt = Infoblox::DNS::Record::TXT->new(
               name    => "txt.domain.com",
               comment => "add TXT txt.domain.com",
               text    => "this is text string",
               );

unless ($bind_txt) {
       die("Construct DNS record TXT failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}

#Add the DNS TXT record object to the Infoblox appliance through a session
$session->add($bind_txt)
       or die("Add record TXT failed: ",
                       $session->status_code() . ":" . $session->status_detail());
print "DNS TXT object added to server successfully\n";
```

#Search for a specific DNS TXT record

```
#Search all TXT records in the zone that matches "domain.com"
my @retrieved_objs = $session->search(
               object => "Infoblox::DNS::Record::TXT",
               name   => 'domain\.com'
               );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS TXT object found at least 1 matching entry\n";

#Search all TXT records with names that start with "txt" and end with "domain.com"
my @retrieved_objs = $session->search(
               object => "Infoblox::DNS::Record::TXT",
               name   => '^txt.*\.domain\.com\$'
               );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS TXT object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS TXT record

```
#Get a TXT record through the session
my @retrieved_objs = $session->get(
               object => "Infoblox::DNS::Record::TXT",
               name   => "txt.domain.com"
               );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS TXT object found at least 1 matching entry\n";

#Modify one of the attributes of the specified TXT record
$object->text("this is new text string");

#Apply the changes
$session->modify($object)
       or die("Modify record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "DNS TXT object modified successfully \n";
```

#Remove a DNS TXT record

```
#Get TXT record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::TXT",
    name   => "txt.domain.com"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record TXT failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS TXT object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "DNS TXT object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session, Infoblox::DNS::View, Infoblox::DNS::Zone

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::RootNameServer - Custom root name server object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DNS::View->custom_root_name_servers( )
  - Infoblox::Grid::Member::DNS->custom_root_name_servers( )
- METHODS
  - host_name( )
  - ipv4addr( )
  - ipv6addr( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::RootNameServer - Custom root name server object.

DESCRIPTION

The custom root name server provides host name and IP address information of the internal root name server. When the Infoblox appliance receives a recursive query for DNS data that it does not have, it uses the internal root name server specified instead of using the Internet root name server.

CONSTRUCTOR

```
my $rootns = Infoblox::DNS::RootNameServer->new(
             host_name => $fqdn,      #Required
             ipv4addr  => $ipv4addr   #Required if $ipv6addr is not present
             ipv6addr  => $ipv4addr   #Required if $ipv4addr is not present
);
```

MODULE METHODS

You can apply the following functions to a custom root name server object:

Infoblox::DNS::View->custom_root_name_servers( )

Use this function to specify custom root name servers at the view level on the Infoblox appliance. See Infoblox::DNS::View->custom_root_name_servers() for parameters and return values.

Example

```
#Construct a custom root name server object
my $rootns1 = Infoblox::DNS::RootNameServer->new(
             host_name => "rns1.test.com",
             ipv4addr  => "4.4.4.4"
);

#Construct a custom root name server object
my $rootns2 = Infoblox::DNS::RootNameServer->new(
             host_name => "rns3.test.com",
             ipv6addr  => "6666::"
);

# Configure custom root name server on the View object
my $response = $View->custom_root_name_servers([$rootns1, $rootns2]);
```

Infoblox::Grid::Member::DNS->custom_root_name_servers( )

Use this function to specify custom root name servers at the member level on the Infoblox appliance. See Infoblox::Grid::Member::DNS->custom_root_name_servers() for parameters and return values.

Example

```
#Construct a custom root name server object
```

```
my $rootns1 = Infoblox::DNS::RootNameServer->new(
            host_name => "rns1.test.com",
            ipv4addr  => "4.4.4.4"
);

#Construct a custom root name server object
my $rootns2 = Infoblox::DNS::RootNameServer->new(
            host_name => "rns3.test.com",
            ipv6addr  => "6666::"
);

# Configure custom root name server on the member DNS object
my $response = $Member_DNS->custom_root_name_servers([$rootns1, $rootns2]);
```

METHODS

This section describes all the methods you can use to configure and retrieve the attribute values of a custom root name server.

host_name( )

Use this method to set or retrieve the host name of the custom root name server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the host name in the FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A host name can have a maximum of 256 characters.

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you do not specify a parameter, the method returns the attribute value.

Example

```
#Get host name
my $host_name = $rootns->host_name();
#Modify host name
$rootns->host_name("qa.infoblox.com");
```

ipv4addr( )

Use this method to set or retrieve the IPv4 address of the custom root name server. It is required only when IPv6 is not specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192. 168.1.2).

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you do not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $rootns->ipv4addr();
#Modify ipv4addr
$rootns->ipv4addr("2.2.2.2");
```

ipv6addr( )

Use this method to set or retrieve the IPv6 address of the custom root name server. It is required only when IPv4 is not specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv6 address is a 128-bit number in colon hexadecimal notation. It consists of eight groups of four hexadecimal digits separated by colons (example: 12ab:0000:0000:0123:4567:89ab:0000:cdef).

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you do not specify a parameter, the method returns the attribute value.

Example

```
# Getting ipv6addr
my $ipv6addr = $rootns->ipv6addr();
# Modifying ipv6addr
$rootns->ipv6addr("12ab::345");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a custom root name server object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a custom root name server object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a custom root name server object

```
my $rootns = Infoblox::DNS::RootNameServer->new(
             host_name => "rns1.test.com",
             ipv4addr  => "4.4.4.4"
);
unless($rootns) {
       die("Construct custom root name server object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Root name server object created successfully\n";
```

#Get the Member DNS object and add custom root name server to it

```
#Get the Member DNS object
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member::DNS",
    name   => "infoblox.localdomain"
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get Member DNS object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get Member DNS object found at least 1 matching entry\n";

#Apply the changes to the Grid DNS object
$object->custom_root_name_servers([$rootns]);

#Apply the changes
$session->modify($object)
    or die("Modify Member DNS object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Member DNS object with custom root name server updated to Infoblox appliance successfully\n";
```

#Modify custom root name server

```
#Modify existing custom root name server object

#Modifying the host name of the custom root name server object.
$rootns->host_name("eng.infoblox.com");
```

```
#Modifying the IPV4 address of the custom root name server object.
$rootns->ipv4addr("3.3.3.3");

#Apply changes to the Member DNS object.
$object->custom_root_name_servers([$rootns]);

#Update Member DNS object through the Infoblox session.
$session->modify($object)
            or die("modify Member DNS object failed: ",
                $session->status_code() . ":" . $session->status_detail());
       print "Member DNS object with modified custom root name server updated to Infoblox appliance
successfully\n";
```

#Remove custom root name server

```
#Apply changes to the Member DNS object to use default internet root name server.
$object->custom_root_name_servers(undef);

#Update Member DNS object through the Infoblox session.
$session->modify($object)
            or die("Remove custom root name server from Member DNS object failed: ",
                $session->status_code() . ":" . $session->status_detail());
       print "Removed custom root name server from Member DNS object successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Grid::Member::DNS, Infoblox::DNS::View

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Ruleset - Ruleset object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
  - type( )
  - comment( )
  - disabled( )
  - nxdomain_rules( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Ruleset - Ruleset object

DESCRIPTION

Represents a Ruleset object, which is a collection of rules that is used to match domain names.

CONSTRUCTOR

See Infoblox::DNS::Ruleset::Rule for details.

```
my $rule1 = Infoblox::DNS::Ruleset::NxdomainRule->new(
    pattern => $string,                                          #Optional
    action => $string,                                           #Optional
);
my $rule2 = Infoblox::DNS::Ruleset::NxdomainRule->new(
    pattern => $string,                                          #Optional
    action => $string,                                           #Optional
);
my $ruleset = Infoblox::DNS::Ruleset->new (
    name => $string,                                             #Required
    type => <"NXDOMAIN" | "BLACKLIST">                            #Required
    comment => $string,                                          #Optional
    disabled => "true" | "false",                                #Optional
    rules => [$rule1, $rule2]                                    #Optional
);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a Ruleset object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Create Rule 1 object
my $rule1 = Infoblox::DNS::Ruleset::NxdomainRule->new(
    pattern => "www.a.zone.com",
    action => 'PASS' );

#Create Rule 2 object
my $rule2 = Infoblox::DNS::Ruleset::NxdomainRule->new(
    pattern=> "www.b.zone.com",
    action => 'REDIRECT' );
```

```
my $ruleset = Infoblox::DNS::Ruleset->new (
   name => $ruleset_name,
   type => "NXDOMAIN",
   comment=> "This is a demo ruleset",
   disabled=> "false",
   nxdomain_rules=> [$rule1, $rule2]);

 #Submit for addition
 my $response = $session->add( $nsg1 );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific Ruleset object:

name - Required. The name of the ruleset in string format.
disabled - Optional. Indicates if the ruleset is disabled. The valid value is either "true" or
"false".
comment - Optional. The comment that was entered for the ruleset.
```

Examples

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Ruleset",
    name   => "Ruleset 1",
    );

my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Ruleset",
    disabled   => "false",
    );

my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Ruleset",
    comment   => "Ruleset comment",
    );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Modify the comment
$ruleset->comment("this is a modified comment");
#Submit the modification
my $response = $session->modify( $ruleset );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the object, and then submit it for removal.

Example

```
#Get the objects with the same name my @retrieved_objs = $session->get( object => "Infoblox::DNS::Ruleset", name =>
"Ruleset 1", ); #Find the desired object on the retrieved list my $ruleset= $retrieved_objs[0]; #Submit the object for removal
my $response = $session->remove( $ruleset );
```

Infoblox::Session->search( )

Use this method to search for Ruleset objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a Ruleset object:

name - Required. The ruleset name in string format (regular expression).
disabled - Optional. Indicates if the ruleset is disabled. The valid value is either "true" or
"false".
comment - Optional. The comment that was entered for the ruleset (regular expression).
```

Examples

```
# Search for all Ruleset objects
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Ruleset",
    name   => ".*",
    );
```

Search for all Ruleset objects with "false" as the value for the "disabled" attribute my @retrieved_objs = $session->get(object => "Infoblox::DNS::Ruleset", disabled => "false");

Search for all Ruleset objects with "Ruleset Comm" as the value for the "comment" attribute my @retrieved_objs = $session->get(object => "Infoblox::DNS::Ruleset", comment => "Ruleset Comm");

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a Ruleset object.

name( )

Use this method to set or retrieve the name of this Ruleset object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The ruleset name in string format, with a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $ruleset->name();
#Modify name
$ruleset->name("rulset");
```

type( )

Use this method to set or retrieve the type of this Ruleset object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The ruleset type. Supported values are "NXDOMAIN" or "BLACKLIST".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get type
my $type = $ruleset->type();
#Modify type
$ruleset->type("NXDOMAIN");
```

comment( )

Use this method to set or retrieve a descriptive comment about the Ruleset object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Comment in string format, with a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $ruleset->comment();
#Modify comment
$ruleset->comment("This is Default Ruleset");
```

disabled( )

Use this method to set or retrieve the flag that indicates if the Ruleset object is disabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disabled flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disabled
my $disabled = $ruleset->disabled();
#Modify disabled
$ruleset->disabled("true");
```

nxdomain_rules( )

Use this method to set or retrieve a list of Infoblox::DNS::Ruleset::NxdomainRule objects assigned to this Ruleset object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains a list of Infoblox::DNS::Ruleset::NxdomainRule objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting rules
my $rules = $ruleset->nxdomain_rules( );
#Modifying rules
$ruleset->nxdomain_rules([$rule1]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as modify and remove. This sample also includes error handling for the operations.

#Preparation prior to a Ruleset object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a Ruleset object

```
my $rule1 = Infoblox::DNS::Ruleset::NxdomainRule->new(
    pattern => "www.a.zone.com",
    action => 'PASS' );
unless ($rule1) {
   die("Construct Rule 1 failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

my $rule2 = Infoblox::DNS::Ruleset::NxdomainRule->new(
    pattern=> "www.b.zone.com",
    action => 'REDIRECT' );
unless ($rule2) {
   die("Construct Rule 2 failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

my $ruleset = Infoblox::DNS::Ruleset->new (
    name => "Ruleset 1",
    type => "NXDOMAIN",
    comment=> "This is demo ruleset",
    disabled=> "false",
    nxdomain_rules=> [$rule1, $rule2]);

unless ($ruleset) {
   die("Construct Ruleset failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Ruleset object created successfully\n";
```

#Add the Ruleset object

```
my $resp = $session->add($ruleset);
unless ($resp == 1) {
   die("Add Ruleset failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
```

#Modify the Ruleset object

```
my @result_array = $session->get(
       object => "Infoblox::DNS::Ruleset",
       name => "Ruleset 1");
if( defined @result_array  and scalar( @result_array ) > 0 ){
       my $ruleset = $result_array[0];
       $ruleset->name("Ruleset 2");

       my $response = $session->modify( $ruleset );
       print "Modify the Ruleset object success.\n" if ( $response );
        }
```

#Remove a Ruleset object

```
my @result_array = $session->get( "object" => "Infoblox::DNS::Ruleset" ,  "name" => "Ruleset 2" );

if( defined @result_array  and scalar( @result_array ) > 0 ){
       my $ruleset = $result_array[0];
       if( ref( $ruleset ) eq "Infoblox::DNS::Ruleset" )
       {

               my $response = $session->remove($ruleset);
               print "Remove Ruleset success.\n" if ( $response );

       }

       unless ($ruleset) {
               die("Remove Ruleset failed: ",
                       $session->status_code() . ":" . $session->status_detail());
       }
        }

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->remove(), Infoblox::Session->modify(), Infoblox::DNS, Infoblox::DNS::Ruleset::NxdomainRule

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Ruleset::NxdomainRule - Rule of Ruleset object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::DNS::Ruleset->nxdomain_rules( )
- METHODS
  - pattern( )
  - action( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Ruleset::NxdomainRule - Rule of Ruleset object

DESCRIPTION

Represents a rule that is used to match domain names.

CONSTRUCTOR

```
my $rule = Infoblox::DNS::Ruleset::NxdomainRule->new(
            pattern => $string,                  #Required
            action => $string,                   #Required
);
```

MODULE METHODS

The following functions can be applied to an Infoblox::DNS::Ruleset::NxdomainRule object.

Infoblox::DNS::Ruleset->nxdomain_rules( )

Use this method to add a list of Infoblox::DNS::Ruleset::NxdomainRule objects to the Ruleset object of an Infoblox appliance. See Infoblox::DNS::Ruleset->nxdomain_rules() for parameters and return values.

Example

```
#Construct a rule my $rule1 = Infoblox::DNS::Ruleset::NxdomainRule->new( pattern => "www.a.zone.com", action => "PASS" );

# Configure a rule for the Ruleset object $ruleset->nxdomain_rules([$rule1]);
```

METHODS

This section describes all the methods you can use to configure and retrieve the attribute values of a Rule object.

pattern( )

Use this method to specify the pattern that is used to match domain names.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The pattern in string format, with a maximum of 1024 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get pattern
my $pattern = $ruleset_item->pattern();
```

```
#Modify pattern
$ruleset_item->pattern("^*$");
```

action( )

Use this method to specify the action to perform when a domain name matches the pattern defined in this Infoblox::DNS::Ruleset::NxdomainRule object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is "PASS" or "REDIRECT".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get action
my $action = $ruleset_item->action();
#Modify action
$ruleset_item->action("PASS");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DNS::Ruleset

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::SharedRecord::A - DNS shared A record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - ipv4addr( )
  - name( )
  - shared_record_group( )
  - ttl( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::SharedRecord::A - DNS shared A record object.

DESCRIPTION

A shared A (address) record is similar to a regular A record. It maps a domain name to an IPv4 address. The difference is that a shared A record should be added into a shared record group. If this shared record group is associated with other zones, then this shared A record will be shared into these zones. For more information about how shared record groups and shared records work, please refer to Infoblox Administrator Guide.

CONSTRUCTOR

```
 my $shared_a = Infoblox::DNS::SharedRecord::A->new (
     ipv4addr                => $ipv4addr,          #Required
     name                    => $string,            #Required
     shared_record_group      => $string,            #Required
     comment                 => $string,            #Optional / Default is empty
     disable                 => "true" | "false"    #Optional / Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl                     => $num | undef,       #Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS shared A object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $shared_a = Infoblox::DNS::SharedRecord::A->new (
    ipv4addr            => "1.1.1.1",
    name                => "shared_a",
    comment             => "this is a demo shared A record",
    disable             => "true",
    ttl                 => "33",
    shared_record_group => "test_srg"  );
# Submit for addition
my $response = $session->add( $shared_a );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS shared A object:

 name - Required. A host name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::A",
    name   => "shared_a");

# get all A shared records with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::A",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$shared_a->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $shared_a );
```

Infoblox::Session->remove( )

Use this method to submit to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::A",
    name   => "shared_a");
# Find the desired object from the retrieved list.
my $desired_a = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_a );
```

Infoblox::Session->search( )

Use this method to search for DNS shared A record objects in the Infoblox appliance. See Infoblox::Session->search() for para meters and return values.

Key References

```
Apply the following attributes to search for a DNS shared A record object:

 name - Required. A host name in string format (regular expression).
 comment - Optional. A comment in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS A objects under domain.com in default view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::A",
    name   => "shared");

# search all DNS shared A records with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::SharedRecord::A",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DNS shared A record object.

comment( )

Use this method to set or retrieve the descriptive comment of a DNS shared A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $shared_a->comment();
#Modify comment
$shared_a->comment("Modifying the DNS A comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS shared A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS shared A record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $shared_a->disable();
#Modify disable
$shared_a->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS shared A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $shared_a->extensible_attributes();
#Modify extensible attributes
$shared_a->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
```

```
});
```

ipv4addr( )

Use this method to set or retrieve the IPv4 address of a DNS shared A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192. 168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $shared_a->ipv4addr();
#Modify ipv4addr
$shared_a->ipv4addr("2.2.2.2");
```

name( )

Use this method to set or retrieve the host name of a DNS shared A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Hostname of this shared A record. This is not the FQDN name of the host name. It should only be the hostname portion of FQDN (Fully Qualified Domain Name) name. For example, if this shared A record 'shared_a' will be shared in zone test.com, then the name should be 'shared_a'. System will append 'test.com' when this shared A record is populated and shared in zone test.com.

A hostname can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $shared_a->name();
#Modify name
$shared_a->name("shared_a");
```

shared_record_group( )

Use this method to set or retrieve the shared record group of a DNS shared A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of defined Infoblox::DNS::SRG objects. The shared A record will be defined under this shared record group and shared among zones associated with this shared record group.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get shared record group
my $srg = $shared_a->shared_record_group();
#Modify shared record group
$shared_a->shared_record_group("test_srg");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value of a DNS shared A record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $shared_a->ttl();
#Modify ttl
$shared_a->ttl(1800);
#Un-override ttl
$shared_a->ttl(undef);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS shared A record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox Appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";

#Create the shared record group prior to an shared A record insertion
my $srg = Infoblox::DNS::SRG->new(name => "test_srg");
unless ($srg) {
   die("Construct srg failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "SRG object created successfully\n";

#Verify if the SRG exists
my $object = $session->get(object => "Infoblox::DNS::SRG", name => "test_srg");
unless ($object) {
   print "SRG does not exist on server, safe to add the SRG\n";
   $session->add($SRG)
      or die("Add SRG failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "SRG added successfully\n";
```

#Create a DNS shared A record

```
#Construct a DNS A object
my $shared_a = Infoblox::DNS::SharedRecord::A->new(
    name     => "shared_a",
    comment  => " this is a demo shared_a record ",
```

```
    ipv4addr => "1.1.1.1",
    shared_record_group => "test_srg"
);
unless ($shared_a) {
   die("Construct DNS record A failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS A object created successfully\n";

#Add the DNS shared A record object to Infoblox Appliance through a session
$session->add($shared_a)
    or die("Add record A failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS A object added to server successfully\n";
```

#Search for a specific DNS shared A record

```
#Search all shared A records
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::A",
    name   => ".*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search shared record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared A object found at least 1 matching entry\n";
```

#Get and modify a DNS shared A record

```
#Get shared A record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::A",
    name   => "shared_a"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get shared record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS shared A object found at least 1 matching entry\n";

#Modify one of the attributes of the specified shared A record
$object->ipv4addr("3.3.3.3");

#Apply the changes
$session->modify($object)
    or die("Modify record A failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS A object modified successfully \n";
```

#Remove a DNS shared A record

```
#Get shared A record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::A",
    name   => "shared_a"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get shared record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS shared A object found at least 1 matching entry\n";


#Submit the object for removal
$session->remove($object)
    or die("Remove shared record A failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "DNS shared A object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::DNS::SRG, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::SharedRecord::AAAA - DNS AAAA shared record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - ipv6addr( )
  - name( )
  - shared_record_group( )
  - ttl( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::SharedRecord::AAAA - DNS AAAA shared record object.

DESCRIPTION

A shared AAAA record is similar to a regular AAAA record. It maps a domain name to an IPv6 address. The difference is that a shared AAAA record should be added into a shared record group. If this shared record group is associated with other zones, then this shared AAAA record will be shared into these zones. For more information about how shared record groups and shared records work, please refer to Infoblox Administrator Guide.

CONSTRUCTOR

```
 my $srg_aaaa = Infoblox::DNS::SharedRecord::AAAA->new (
     ipv6addr                => $ipv6addr,          #Required
     name                    => $fqdn,              #Required
     shared_record_group      => $string,            #Required
     comment                 => $string,            #Optional / Default is empty
     disable                 => "true" | "false"    #Optional / Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl                     => $num | undef,       #Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS AAAA shared record object.

Infoblox::Session->add( )

Use this method to add a AAAA shared record object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $srg_aaaa = Infoblox::DNS::SharedRecord::AAAA->new(
    name     => "aaaa1",
    comment  => "add a shared DNS AAAA aaaa1",
    ipv6addr => "aaaa:1111:bbbb:2222:cccc:3333:dddd:4444",
    disable  => "true",
    ttl      => "33",
    shared_record_group => "test_srg"  );
# Submit for addition
my $response = $session->add( $srg_aaaa );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS AAAA object:

 name - Required. A host name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
                     object => "Infoblox::DNS::SharedRecord::AAAA",
                     name   => "aaaa1");

# get all shared AAAA records with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::AAAA",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a AAAA shared record object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$srg_aaaa->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $srg_aaaa );
```

Infoblox::Session->remove( )

Use this method to remove a AAAA shared record object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific AAAA shared record object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::AAAA",
    name   => "aaaa1");
# Find the desired object from the retrieved list.
my $desired_aaaa = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_aaaa );
```

Infoblox::Session->search( )

Use this method to search for DNS AAAA shared record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes, to search for a DNS AAAA shared record object:

 name    - Required. A host name in string format (regular expression).
 comment - Optional. Comment in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS AAAA objects
my @retrieved_objs = $session->search(
    object  => "Infoblox::DNS::SharedRecord::AAAA",
    name    => ".*",
    comment => ".*modified comment" );

# search all DNS shared AAAA records with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
```

```
object => "Infoblox::DNS::SharedRecord::AAAA",
extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DNS shared AAAA record object.

comment( )

Use this method to set or retrieve the descriptive comment of a DNS shared AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $srg_aaaa->comment();
#Modify comment
$srg_aaaa->comment("Modifying the DNS AAAA comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS shared AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS shared AAAA record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $srg_aaaa->disable();
#Modify disable
$srg_aaaa->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS shared AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $srg_aaaa->extensible_attributes();
#Modify extensible attributes
$srg_aaaa->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

ipv6addr( )

Use this method to set or retrieve the IPv6 address of a DNS shared AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv6 address is a 128-bit number in colon hexadecimal notation. It consists of eight 16-bit groups of hexadecimal digits separated by colons (example: 12ab:0000:0000:0123:4567:89ab:0000:cdef).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6addr
my ipv6addr = $srg_aaaa->ipv6addr();
#Modify ipv6addr
$srg_aaaa->ipv6addr("aaaa:1111:bbbb:2222:cccc:3333:dddd:5555");
```

name( )

Use this method to set or retrieve the host name of a DNS shared AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Hostname of this shared AAAA record. This is not the FQDN name of the host name. It should only be the hostname portion of FQDN (Fully Qualified Domain Name) name. For example, if this shared AAAA record 'shared_aaaa' will be shared in zone test.com, then the name should be 'shared_a'. System will append 'test.com' when this shared AAAA record is populated and shared in zone test.com.

A hostname can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $shared_aaaaa->name();
#Modify name
$shared_aaaaa->name("shared_aaaa");
```

shared_record_group( )

Use this method to set or retrieve the shared record group of a DNS shared AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of defined Infoblox::DNS::SRG objects. The shared AAAA record will be defined under this shared record group and shared among zones associated with this shared record group.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get shared record group
my $srg = $shared_aaaa->shared_record_group();
#Modify shared record group
$shared_aaaa->shared_record_group("test_srg");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value of a DNS shared AAAA record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $srg_aaaa->ttl();
#Modify ttl
$srg_aaaa->ttl(1800);
#Un-override ttl
$srg_aaaa->ttl(undef);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a shared AAAA record object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS shared AAAA shared record insertion

```
#PROGRAM STARTS: Include all the modules that will be used

use strict;
use Infoblox;

#Create a session to the Infoblox Appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2", #appliance host ip
    username => "admin",     #appliance user login
    password => "infoblox"   #appliance password
);

unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}

#Create the shared record group prior to an shared AAAA record insertion
my $srg = Infoblox::DNS::SRG->new(name => "test_srg");
unless ($srg) {
   die("Construct srg failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "SRG object created successfully\n";

#Verify if the SRG exists
my $object = $session->get(object => "Infoblox::DNS::SRg", name => "test_srg");
unless ($object) {
   print "SRG does not exist on server, safe to add the SRG\n";
   $session->add($SRG)
      or die("Add SRG failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "SRG added successfully\n";
```

#Create a DNS AAAA shared record

```
#Construct a DNS AAAA object

my $srg_aaaa = Infoblox::DNS::SharedRecord::AAAA->new(
    name     => "aaaa1",
    comment  => "add a DNS shared AAAA aaaa1",
    ipv6addr => "aaaa:1111:bbbb:2222:cccc:3333:dddd:4444",
    shared_record_group => "test_srg"
);

unless ($srg_aaaa) {
   die("Construct DNS shared record AAAA failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS shared AAAA object created successfully\n";

#Add the DNS AAAA shared record object to Infoblox Appliance through a session
$session->add($srg_aaaa)

    or die("Add record AAAA failed: ",
           $session->status_code() . ":" . $session->status_detail());
     print "DNS AAAA object added to server successfully\n";
```

#Search for a specific DNS AAAA shared record

```
#Search all AAAA shared records in the zone
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::AAAA",
    name   => ".*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search shared record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared AAAA object found at least 1 matching entry\n";

#Search all AAAA shared records that start with "add" in the comment
my @retrieved_objs = $session->search(
    object    => "Infoblox::DNS::SharedRecord::AAAA",
    comment   => "add.*",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search shared record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared AAAA object found at least 1 matching entry\n";

#Search all shared AAAA shared records that start with "aaaa1"
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::AAAA",
    name   => "^aaaa1",
 );
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search shared record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared AAAA object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS AAAA shared record

```
#Get AAAA shared record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::AAAA",
    name   => "aaaa1"
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get shared record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

#Modify one of the attributes of the specified AAAA shared record
```

```
$object->ipv6addr("aaaa:1111:bbbb:2222:cccc:3333:dddd:eeee");

#Applying the changes
$session->modify($object)

    or die("Modify shared record AAAA failed: ",
           $session->status_code() . ":" . $session->status_detail());
     print "DNS shared AAAA object modified successfully \n";
```

#Remove a DNS AAAA shared record.

```
#Get AAAA shared record through the session
my @retrieved_objs = $session->get(
    object   => "Infoblox::DNS::SharedRecord::AAAA",
    name     => "aaaa1"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get record AAAA failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

print "Get DNS shared AAAA object found at least 1 matching entry\n";

#Submit the object for removal

$session->remove($object)
    or die("Remove shared record AAAA failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS shared AAAA object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::DNS::SRG, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::SharedRecord::MX - DNS shared MX record object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - exchanger( )
  - extensible_attributes( )
  - name( )
  - pref( )
  - shared_record_group( )
  - ttl( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::SharedRecord::MX - DNS shared MX record object

DESCRIPTION

A shared MX (mail exchanger) record is similiar to a regular MX record, which maps a domain name to a mail exchanger. The difference is that a shared MX record should be added into a shared record group. If this shared record group is associated with other zones, then this shared MX record will be shared into these zones. For more information about how shared record groups and shared records work, please refer to Infoblox Administrator Guide.

CONSTRUCTOR

```
 my $mx = Infoblox::DNS::SharedRecord::MX->new(
     exchanger               => $fqdn,               #Required
     name                    => $fqdn,               #Required
     pref                    => $num,                #Required
     shared_record_group      => $string,             #Required
     comment                 => $string,             #Optional / Default is empty
     disable                 => "true" | "false",     #Optional / Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl                     => $num | undef         #Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS shared MX object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $srg_mx = Infoblox::DNS::SharedRecord::MX->new(
    name      => "bind_mx",
    comment   => "add mx bind_mx",
    pref      => 1,
    exchanger => "exchanger",
);
# Submit for addition
```

```
my $response = $session->add( $srg_mx );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS shared MX object:

 name - Required.  A domain name in string format
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
 my @retrieved_objs = $session->get(
    object    => "Infoblox::DNS::SharedRecord::MX",
    name      => "bind_mx");

# get all shared MX records with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::MX",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the comment.
$srg_mx->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $srg_mx );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my @retrieved_objs = $session->get(
    name      => "bind_mx",
    comment   => "add mx bind_mx",
    pref      => 1,
    exchanger => "exchanger");
# find the desired object from retrieved list.
my $desired_mx = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_mx );
```

Infoblox::Session->search( )

Use this method to search for DNS shared MX record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS shared MX record object:

 name - Required. A domain name in string format (regular expression).
 comment - Optional. A comment in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS shared MX objects
    my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::MX",
    name   => ".*");

# search all DNS shared MX records with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::SharedRecord::MX",
```

```
extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DNS shared MX record object.

comment( )

Use this method to add or retrieve a descriptive comment of a DNS shared MX record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for this record in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting comment
my $comment = $srg_mx->comment();
# Modifying comment
$srg_mx->comment("add mx2");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS shared MX record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS shared MX record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting disable
my $disable = $srg_mx->disable()
# Modifying disable
$srg_mx->disable("true");
```

exchanger( )

Use this method to set or retrieve the exchanger of a DNS shared MX record.

The name of the mail exchanger in Fully Qualified Domain Name (FQDN) format.

Parameter

Text with the name of the mail exchanger for the DNS shared MX record.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.
If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting exchanger
my $exchanger = $srg_mx->exchanger();
# Modifying exchanger
$srg_mx->exchanger("exchanger2");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS shared MX record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $srg_mx->extensible_attributes();
#Modify extensible attributes
$srg_mx->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the domain name of a DNS shared MX record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name for the DNS shared MX record.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting name
my $name = $srg_mx->name();
# Modifying name
$srg_mx->name("mx2");
```

pref( )

Use this method to set or retrieve the Preference value of a DNS shared MX record.

The preference value of the DNS shared MX object. Preference value of shared MX record for the zone range from 0 to 65535. Low values are more preferred.

Parameter

Enter an unsigned integer between 0-65535. The priority determines the order in which a client attempts to contact the target mail exchanger. The highest priority is 0 and is queried first.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting pref
my $pref = $srg_mx->pref();
# Modifying pref
$srg_mx->pref(3);
```

shared_record_group( )

Use this method to set or retrieve the shared record group of a DNS shared MX record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of defined Infoblox::DNS::SRG objects. The shared MX record will be defined under this shared record group and shared among zones associated with this shared record group.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get shared record group
my $srg = $shared_mx->shared_record_group();
#Modify shared record group
$shared_mx->shared_record_group("test_srg");
```

ttl( )

Use this method to configure or retrieve the Time to Live (TTL) value of a DNS shared MX record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $srg_mx->ttl();
#Modify ttl
$srg_mx->ttl(1800);
#Un-override ttl
$srg_mx->ttl(undef);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS shared MX record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the shared record group prior to an shared MX record insertion
my $srg = Infoblox::DNS::SRG->new(name => "test_srg");
unless ($srg) {
   die("Construct srg failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
```

```
print "SRG object created successfully\n";

#Verify if the SRG exists
my $object = $session->get(object => "Infoblox::DNS::SRG", name => "test_srg");
unless ($object) {
   print "SRG does not exist on server, safe to add the SRG\n";
   $session->add($SRG)
      or die("Add SRG failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "SRG added successfully\n";
```

#Create a DNS shared MX record

```
#Construct a DNS shared MX object
my $srg_mx = Infoblox::DNS::SharedRecord::MX->new(
    name      => "bind_mx",
    comment   => "add mx bind_mx",
    pref      => 1,
    exchanger => "exchanger",
    shared_record_group => "test_srg"
);

unless ($srg_mx) {
   die("Construct DNS shared record MX failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS shared MX object created successfully\n"; #Add the DNS shared MX record object to Infoblox
appliance through a session
$session->add($srg_mx)
    or die("Add shared record MX failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS shared MX object added to server successfully\n";
```

#Search for a specific DNS shared MX record

```
#Search all shared MX records in the zone
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::MX",
    name   => ".*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search shared record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared MX object found at least 1 matching entry\n";

#Search all shared MX records that start with "bind" in the zone
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::MX",
    name   => "bind.*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search shared record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared MX object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS shared MX record

```
#Get shared MX record through the session
my  @retrieved_objs = $session->get(
    object    => "Infoblox::DNS::SharedRecord::MX",
    name      => "bind_mx",
    pref      => 1,
    exchanger => "exchanger",
);

my $object = $retrieved_objs[0];
unless ($object) {
    die("Get shared record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS shared MX object found at least 1 matching entry\n";
```

```
#Modify one of the attributes of the specified shared MX record
$object->exchanger("exc");

#Apply the changes
$session->modify($object)
    or die("Modify shared record MX failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS shared MX object modified successfully \n";
```

#Remove a DNS shared MX record

```
 #Get shared MX record through the session
 my @retrieved_objs = $session->get(
     object => "Infoblox::DNS::SharedRecord::MX",
     name   => "bind_mx"
  );
   my $object = $retrieved_objs[0];
   unless ($object) {
    die("Get shared record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
  }
  print "Get DNS shared MX object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove shared record MX failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "DNS shared MX object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::DNS::SRG, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::SharedRecord::SRV - DNS shared SRV record object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - name( )
  - port( )
  - priority( )
  - target( )
  - shared_record_group( )
  - ttl( )
  - weight( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::SharedRecord::SRV - DNS shared SRV record object

DESCRIPTION

A shared SRV record is similar to a regular SRV record, which is also known as a service record. A SRV record provides information on available services. The difference is that shared SRV records should be added into a shared record group. If this shared record group is associated with other zones, then this shared SRV record will be shared into these zones. For more information about how shared record groups and shared records work, please refer to Infoblox Administrator Guide.

CONSTRUCTOR

```
 my $srv = Infoblox::DNS::SharedRecord::SRV->new(
     name                    => $string,              #Required
     port                    => $num,                 #Required
     priority                => $num,                 #Required
     shared_record_group      => $string,              #Required
     target                  => $fqdn,                #Required
     weight                  => $num,                 #Required
     comment                 => $string,              #Optional / Default is empty
     disable                 => "true" | "false"      #Optional / Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl                     => $num | undef,         #Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a DNS shared SRV record object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct object
my $srg_srv = Infoblox::DNS::SharedRecord::SRV->new(
    name     => "srg_srv",
    comment  => "add shared srv srg_srv",
```

```
        priority => 1,
        port     => 1,
        weight   => 1,
        target   => "old.target.test.org",
        shared_record_group => "test_srg"
    );
    # Submit for addition
    my $response = $session->add( $srg_srv );
```

Infoblox::Session->get( )

Use this method to retrieve the existing objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get a specified DNS shared SRV object:

 name - Required. A name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::SRV",
    name     => "srg_srv");

# get all shared SRV records with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::SRV",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$srg_srv->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $srg_srv);
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::SRV",
    name     => "srg_srv");
my $desired_srv= $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_srv );
```

Infoblox::Session->search( )

Use this method to search for existing DNS shared SRV record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes, to search for a DNS shared SRV record object:

  name    - Required. A name of shared SRV record (regular expression).
  comment - Optional . A comment in string format (regular expression).
  extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS shared SRV object with comment containing 'shared'.
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::SRV",
    name     => ".*",
```

```
    comment  => "shared");

# search all DNS shared SRV records with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::SharedRecord::SRV",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DNS shared SRV record object.

comment( )

Use this method to add or retrieve a descriptive comment of a DNS shared SRV record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get comment
my $comment = $sharedsrv->comment();
#Modify comment
$sharedsrv->comment("Modifying the DNS shared SRV comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS shared SRV record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false, therefore the DNS shared SRV record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get disable
my $disable = $sharedsrv->disable();
#Modify disable
$sharedsrv->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS shared SRV record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $sharedsrv->extensible_attributes();
#Modify extensible attributes
$sharedsrv->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

name( )

Use this method to set or retrieve the name of a DNS shared SRV record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the DNS shared SRV record.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get name
my $name = $sharedsrv->name();
#Modify name
$sharedsrv->name("modified_srg_srv");
```

port( )

Use this method to add or retrieve a port of a DNS shared SRV record.

Parameter

The port number for the DNS shared SRV record (range from 0 to 65535).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get port
my $port = $sharedsrv->port();
#Modify port
$sharedport->port("2");
```

priority( )

Use this method to add or retrieve the priority of a DNS shared SRV record.

Parameter

Number with the priority for the DNS shared SRV record (range from 0 to 65535).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get priority
my $priority = $sharedsrv->priority();
#Modify priority
$sharedsrv->priority("2");
```

target( )

Parameter

Text with the target for the DNS shared SRV record.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get target
```

```
my $target = $sharedsrv->target();
#Modify target
$sharedsrv->target ("new.target.test.org");
```

shared_record_group( )

Use this method to set or retrieve the shared record group of a DNS shared SRV record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of defined Infoblox::DNS::SRG objects. The shared SRV record will be defined under this shared record group and shared among zones associated with this shared record group.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get shared record group
my $srg = $shared_srv->shared_record_group();
#Modify shared record group
$shared_srv->shared_record_group("test_srg");
```

ttl( )

Use this method to configure or retrieve the Time to Live (TTL) value of a DNS shared SRV record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined; therefore the record inherits the TTL value from the zone level.

With a specified TTL value, the method overrides the zone values with the specified TTL value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get ttl
my $ttl = $sharedsrv->ttl();
#Modify ttl
$sharedsrv->ttl(1800);
#Un-override ttl
$sharedsrv->ttl(undef);
```

weight( )

Use this method to set or retrieve the weight of a DNS shared SRV record.

Parameter

The weight of the DNS shared SRV record.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get weight
my $weight = $sharedsrv->weight ();
#Modify weight
$sharedsrv->weight ("3");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, search, modify, and remove an object. This sample code also includes error handling for the operations.

#Preparation prior to a DNS shared SRV record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";

#Create the shared record group prior to an shared SRV record insertion
my $srg = Infoblox::DNS::SRG->new(name => "test_srg");
unless ($srg) {
   die("Construct srg failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "SRG object created successfully\n";

#Verify if the SRG exists
my $object = $session->get(object => "Infoblox::DNS::SRG", name => "test_srg");
unless ($object) {
   print "SRG does not exist on server, safe to add the SRG\n";
   $session->add($SRG)
      or die("Add SRG failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "SRG added successfully\n";
```

#Create a DNS shared SRV record

```
my $srg_srv = Infoblox::DNS::SharedRecord::SRV->new(
   name     => "srg_srv",
   comment  => "add shared srg_srv",
   priority => 1,
   port     => 1,
   weight   => 1,
   target   => "old.target.test.org",
   shared_record_group => "test_srg",
 );
 unless ($srg_srv) {
    die("Construct DNS shared record SRV failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
 }
print "DNS shared SRV object created successfully\n";

#Add the DNS shared SRV record object to Infoblox  appliance through a session
$session->add($srg_srv)
    or die("Add shared record SRV failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS shared SRV object added to server successfully\n";
```

#Search for a specific DNS shared SRV record

```
#Search for SRV record
my @retrieved_objs = $session->search(
    "object" => "Infoblox::DNS::SharedRecord::SRV",
    "name" => "srg_srv",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search shared record SRV failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared SRV object found at least 1 matching entry\n";

#Search all shared SRV records that  has "shared" in the comment
```

```
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::SRV",
    name   => ".*",
    comment => "shared"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search record SRV failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared SRV object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS shared SRV record

```
#Get a SRV shared record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::SRV",
    name   => "srg_srv",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get shared record SRV failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS shared SRV object found at least 1 matching entry\n";

#Modify one of the attributes of the specified SRV record
$object->comment ("modified comment for shared SRV rec");

#Apply the changes
$session->modify($object)
    or die("Modify shared record SRV failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS shared SRV object modified successfully \n";
```

#Remove a DNS shared SRV record

```
#Get a SRV record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::SRV",
    name   => "srg_srv",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get shared record SRV failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS shared SRV object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove shared record SRV failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS shared SRV object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::DNS::SRG, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::SharedRecord::TXT - DNS shared TXT record object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disable( )
  - extensible_attributes( )
  - name( )
  - text( )
  - shared_record_group( )
  - ttl( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::SharedRecord::TXT - DNS shared TXT record object.

DESCRIPTION

A shared TXT record is similar to a regular TXT record. It contains supplemental information for a host. SPF (Sender Policy Framework) records are specialized TXT records that identify the servers that send mail from a domain. The difference is that shared TXT records should be added into a shared record group. If this shared record group is associated with other zones, then this shared TXT record will be shared into these zones. For more information about how shared record groups and shared records work, please refer to Infoblox Administrator Guide.

CONSTRUCTOR

```
 my $txt = Infoblox::DNS::SharedRecord::TXT->new(
     name                   => $fqdn,            #Required
     text                   => $string           #Required
     shared_record_group     => $string,          #Required
     comment                => $string,          #Optional / Default is empty
     disable                => "true" | "false"   #Optional / Default "false"
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ttl                    => $num | undef,      #Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a TXT shard record object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the TXT object
my $srg_txt = Infoblox::DNS::SharedRecord::TXT->new(
    name    => "txt",
    text    => "this is text string",
    comment => "add TXT txt",
    disable => "false",
    ttl     => "360",
    shared_record_group => "test_srg" );
# Submit for addition
my $response = $session->add( $srg_txt );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS shared TXT object:

 name - Required. Host name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::TXT",
    name   => "txt");

# get all shared TXT records with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::TXT",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$srg_txt->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $srg_txt );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specifc object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::TXT",
    name   => "txt");

# find the desired object from the retrieved list.
my $desired_txt = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $ desired_txt );
```

Infoblox::Session->search( )

Use this method to search for DNS shared TXT record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS shared TXT record object:

 name    - Required. The name of TXT record (regular expression).
 comment - Optional. A comment in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS shared TXT objects.
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SharedRecord::TXT",
    name   => ".*");

# search all DNS shared TXT records with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::SharedRecord::TXT",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DNS shared TXT record object.

comment( )

Use this method to set or retrieve the descriptive comment of a DNS shared TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $srg_txt->comment();
#Modify comment
$srg_txt->comment("Modifying the DNS shared TXT comment");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS shared TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The DNS shared TXT record is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $srg_txt->disable();
#Modify disable
$srg_txt->disable("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS shared TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $srg_txt->extensible_attributes();
#Modify extensible attributes
```

```
$srg_txt->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
});
```

name( )

Use this method to set or retrieve the name of a DNS shared TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the DNS shared TXT object. The name can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $srg_txt->name();
#Modify name
$srg_txt->name("new");
```

text( )

Use this method to set or retrieve the text of a DNS shared TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text that you want to associate with the record. It can contain up to 255 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $srg_txt->text();
#Modify name
$srg_txt->text("this is new text string");
```

shared_record_group( )

Use this method to set or retrieve the shared record group of a DNS shared TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of defined Infoblox::DNS::SRG objects. The shared TXT record will be defined under this shared record group and shared among zones associated with this shared record group.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get shared record group
my $srg = $shared_txt->shared_record_group();
#Modify shared record group
$shared_txt->shared_record_group("test_srg");
```

ttl( )

Use this method to set or retrieve the Time to Live (TTL) value of a DNS shared TXT record.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is undefined which indicates that the record inherits the TTL value of the zone.

Specify a TTL value to override the TTL value at the zone level.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the duration in seconds that the record is cached. Zero indicates that the record should not be cached.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ttl
my $ttl = $srg_txt->ttl();
#Modify ttl
$srg_txt->ttl(1800);
#Un-override ttl
$srg_txt->ttl(undef);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS shared TXT record insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create the shared record group prior to an shared TXT record insertion
my $srg = Infoblox::DNS::SRG->new(name => "test_srg");
unless ($srg) {
   die("Construct srg failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "SRG object created successfully\n";

#Verify if the SRG exists
my $object = $session->get(object => "Infoblox::DNS::SRG", name => "test_srg");
unless ($object) {
   print "SRG does not exist on server, safe to add the SRG\n";
   $session->add($SRG)
      or die("Add SRG failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "SRG added successfully\n";
```

#Create a DNS shared TXT record

```
my $srg_txt = Infoblox::DNS::SharedRecord::TXT->new(
              name    => "txt",
              comment => "add TXT txt",
              text    => "this is text string",
              shared_record_group => "test_srg",
              );
```

```
unless ($srg_txt) {
       die("Construct DNS shared record TXT failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}

#Add the DNS shared TXT record object to the Infoblox appliance through a session
$session->add($srg_txt)
       or die("Add shared record TXT failed: ",
                       $session->status_code() . ":" . $session->status_detail());
print "DNS shared TXT object added to server successfully\n";
```

#Search for a specific DNS shared TXT record

```
#Search all shared TXT records in the zone
my @retrieved_objs = $session->search(
               object => "Infoblox::DNS::SharedRecord::TXT",
               name   => ".*"
               );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search shared record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared TXT object found at least 1 matching entry\n";

#Search all TXT records whose name starts with "txt".
my @retrieved_objs = $session->search(
               object => "Infoblox::DNS::SharedRecord::TXT",
               name   => "txt.*"
               );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search shared record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS shared TXT object using regexp found at least 1 matching entry\n";
```

#Get and modify a DNS shared TXT record

```
#Get a shared TXT record through the session
my @retrieved_objs = $session->get(
               object => "Infoblox::DNS::SharedRecord::TXT",
               name   => "txt"
               );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS shared TXT object found at least 1 matching entry\n";

#Modify one of the attributes of the specified TXT record
$object->text("this is new text string");

#Apply the changes
$session->modify($object)
       or die("Modify shared record TXT failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "DNS shared TXT object modified successfully \n";
```

#Remove a DNS shared TXT record

```
#Get TXT record through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SharedRecord::TXT",
    name   => "txt"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get shared record TXT failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS shared TXT object found at least 1 matching entry\n";
```

```
#Submit the object for removal
$session->remove($object)
       or die("Remove shared record TXT failed: ",
              $session->status_code() . ":" . $session->status_detail());
print "DNS shared TXT object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::DNS::SRG, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Sortlist - DNS Sortlist Object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DNS->sortlist( )
  - Infoblox::Grid::Member::DNS->sortlist( )
- METHODS
  - match_list( )
  - source_ipv4addr( )
  - source_ipv6addr( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Sortlist - DNS Sortlist Object.

DESCRIPTION

A sortlist sorts the order of addresses in responses made to DNS queries.

CONSTRUCTOR

```
 my $sort_list = Infoblox::DNS::Sortlist->new(
     source_ipv4addr => $ipv4addr | $ipv4_subnet/prefix
#Either source_ipv4addr or source_ipv6addr is required
     source_ipv6addr => $ipv6addr | $ipv6_subnet/prefix
#Either source_ipv4addr or source_ipv6addr is required
     match_list      => [ $ipv4addr | $ipv4_subnet/prefix | $ipv6addr |  $ipv6_subnet/prefix ...] | undef
#Optional / Default is undefined
 );
```

MODULE METHODS

The following functions are available to apply to an Sortlist object.

Infoblox::Grid::DNS->sortlist( )

Use this function to specify a sortlist at the grid level on the Infoblox appliance. See Infoblox::Grid::DNS->sortlist() for parameters and return values.

Example

```
# Construct the object
my $sortlist = Infoblox::DNS::Sortlist->new(
    match_list      => ["10.20.1.0/24", "10.20.2.0/24"],
    source_ipv4addr => "1.2.3.4"
);
# Configure sortlist on the grid DNS object
my $response = $Grid_DNS->sortlist([$sortlist]);
```

Infoblox::Grid::Member::DNS->sortlist( )

Use this function to specify a sortlist at the grid member level. See Infoblox::Grid::Member::DNS->sortlist() for parameters and return values.

Example

```
# Construct the object
my $sortlist = Infoblox::DNS::Sortlist->new(
    match_list      => ["10.20.1.0/24", "10.20.2.0/24"],
    source_ipv4addr => "1.2.3.4"
);
# Configure sortlist on a Grid Member DNS object
my $response = $Grid_Member_DNS->sortlist([$sortlist]);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a sortlist object.

match_list( )

Use this method to set or retrieve the match list of a sortlist.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference which contains IP addresses and/or networks.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get match list.
my $ref_matchlist = $sort_list->match_list();
#Modify match list.
$sort_list->match_list(["10.0.0.10", "20.0.0.0/24", "2001:db8:1234::/48", "2001:db8::1428:57ab"]);
```

source_ipv4addr( )

Use this method to set or retrieve the source IPv4 address of a sortlist object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Address or network of querier which will be preferred by the resolver as the result of a query. To specify "ANY" querier, please use "0.0.0.0/0".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addrs
my $source_ipv4addr = $sort_list->source_ipv4addr();
#Modify ipv4addrs
$sort_list->source_ipv4addr("10.20.1.0/24");
```

source_ipv6addr( )

Use this method to set or retrieve the source IPv6 address of a sortlist object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Address or network of querier which will be preferred by the resolver as the result of a query.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6addrs
my $source_ipv6addr = $sort_list->source_ipv6addr();
#Modify ipv6addrs
$sort_list->source_ipv6addr("2001:25d8:44:55::/64");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, modify. This sample also

includes error handling for the operations.

#Preparation prior to a DNS sortlist object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
          master   => "192.168.1.2",
          username => "admin",
          password => "infoblox"
);

unless ($session) {
   die("Construc session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a Sortlist object

```
my $sort1 = Infoblox::DNS::Sortlist->new(
    match_list      => ["10.20.1.0/24", "10.20.2.0/24"],
    source_ipv4addr => "1.2.3.4"
);

unless($sort1) {
     die("Construct sort list failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "First sort list object (IPv4) created successfully\n";
```

#Create another sortlist to demonstrate IPv6

```
my $sort2 = Infoblox::DNS::Sortlist->new(
    match_list      => ["2001:db8:1234::/48", "2001:db8:345::/64", "192.168.1.1"],
    source_ipv6addr => "2001:db8::1428:57ab"
);

unless($sort2) {
     die("Construct sort list failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Second sort list object (IPv6) created successfully\n";
```

#Get the Member DNS object to add sortlist object to it

```
my @result_array = $session->get(
    object => "Infoblox::Grid::Member::DNS",
    name   => $host_name
);

unless (scalar(@result_array) == 0) {
    my $memberdns = $result_array[0];

    if ($memberdns) {

        #Apply changes to the Member object.
        $memberdns->sortlist([$sort1, $sort2])
            or die("modify member failed: ",
                   $session->status_code() . ":" . $session->status_detail());
        print "sort lists added to Member DNS object successfully\n";

        #Update member DNS object through the Infoblox session.
        $session->modify($memberdns)
            or die("modify session failed: ",
                   $session->status_code() . ":" . $session->status_detail());
        print "Member DNS object with sortlists updated to Infoblox device successfully\n";

        #Modify existing sortlist objects

        #Modifying the value of the specified object.
        $sort1->source_ipv4addr("10.20.1.0/24");
        print "Modify first sortlist value\n";
```

```
        #Modifying the value of the specified object.
        $sort1->match_list(["10.0.0.10", "20.0.0.0/24", "30.0.0.30", "40.10.0.0/24"]);

        #Modifying the value of the specified object.
        $sort2->source_ipv6addr("2001:db8::1428:0/112");
        print "Modify second sortlist value\n";

        #Apply changes to the member object.
        $memberdns->sortlist([$sort1, $sort2])
            or die("modify member failed: ",
                   Infoblox::status_code() . ":" . Infoblox::status_detail());
        print "sort list updated to Member DNS object successfully\n";

        #Update member DNS object through the Infoblox session.
        $session->modify($memberdns)
            or die("modify session failed: ",
                $session->status_code() . ":" . $session->status_detail());
                print "Member DNS object with sortlist updated to Infoblox device successfully\n";
             }
         } else {
             print "No member found with the specified name.";
         }

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Grid::Member::DNS->sortlist(), Infoblox::Grid::DNS->sortlist()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::SRG - Create a shared record group.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - extensible_attributes( )
  - name( )
  - override_record_name_policy( )
  - record_name_policy( )
  - zone_association( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::SRG - Create a shared record group.

DESCRIPTION

A shared record group (SRG) is created to contain DNS shared records and share them between different zones. For example, if a group of DNS records needs to be in three different zones, you can include the records in a shared record group and assign the group to the three zones. For more information about shared record groups and shared records, please refer to Infoblox Administrator Guide.

CONSTRUCTOR

```
 my $srg = Infoblox::DNS::SRG->new(
     name                        => $string,               #Required
     comment                     => $string,               #Optional / Default is undefined
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     override_record_name_policy => "true" | "false",      #Optional / Default is "false"
     record_name_policy          => $string,               #Optional / Required when
override_record_name_policy is set to "true". Default is undefined
     zone_association            => [$Zone1, $Zone2, ...]  #Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to a DNS SRG object.

Infoblox::Session->add( )

Use this method to add an SRG object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $srg = Infoblox::DNS::SRG->new (
    name     => "test_srg",
    comment  => "this is a demo SRG object",
    );
# Submit for addition
my $response = $session->add( $srg );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS SRG object:

 name - Required. A SRG name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SRG",
    name   => "test_srg");

# get all shared record group objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SRG",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$srg->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $srg );
```

Infoblox::Session->remove( )

Use this method to submit to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SRG",
    name   => "test_srg");
# Find the desired object from the retrieved list.
my $desired_srg = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_srg );
```

Infoblox::Session->search( )

Use this method to search for DNS SRG record objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS SRG record object:

 name - Required. A SRG name in string format (regular expression).
 comment - Optional. A comment in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all DNS SRG objects with name containing 'test'
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SRG",
    name   => ".*test.*");

# search all DNS SRG objects with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::SRG",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DNS SRG object.

comment( )

Use this method to set or retrieve the descriptive comment of a DNS SRG object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $srg->comment();
#Modify comment
$srg->comment("Modifying the DNS SRG comment");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS SRG object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $srg->extensible_attributes();
#Modify extensible attributes
$srg->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the shared record group name of a DNS SRG object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of shared record group.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $srg->name();
#Modify name
$srg->name("test_srg");
```

override_record_name_policy( )

Use this method to set or retrieve the override record name policy flag of a DNS SRG object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false, which means this SRG will use default record name policy setting defined at grid level.

Parameter

Specify "true" to set the override_record_name_policy flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_record_name_policy
my $override = $srg->doverride_record_name_policy();
#Modify override_record_name_policy
$srg->override_record_name_policy("true");
```

record_name_policy( )

Use this method to set or retrieve the record name policy of a DNS SRG object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the record name policy defined at the grid level. See Infoblox::Grid::DNS for more information about record name policies.

This property is required only if override_record_name_policy is set to "true", which means SRG object will use its own record name policy.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get record name policy
my $policy = $srg->record_name_policy();
#Modify record name policy
$srg->record_name_policy("Allow Any");
```

zone_association( )

Use this method to set or retrieve the zone association of a DNS SRG object. All zone objects in a association list are associated with the SRG object, which means that all shared records in this SRG will be shared among these zones. For more information about SRG and zone association, please refer to Infoblox Administrator's Guide.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference of defined Infoblox::DNS::Zone objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get zone association
my $zone_associations = $srg->zone_association();
#Modify zone associations, list of Infoblox::DNS::Zone objects
$srg->zone_association([$zone1,$zone2,$zone3]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a DNS SRG object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
```

```
use strict;
use Infoblox;

#Create a session to the Infoblox Appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";

#Create the zone prior to an SRG object insertion
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
   die("Construct zone failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Zone object created successfully\n";

#Verify if the zone exists
my $object = $session->get(object => "Infoblox::DNS::Zone", name => "domain.com");
unless ($object) {
   print "Zone does not exist on server, safe to add the zone\n";
   $session->add($zone)
      or die("Add zone failed: ",
             $session->status_code() . ":" . $session->status_detail());
}
print "Zone added successfully\n";
```

#Create a DNS SRG object

```
#Construct a DNS SRG object
my $srg = Infoblox::DNS::SRG->new(
    name     => "test_srg",
    comment  => "this is a demo srg object",
    zone_association=> [ $zone ],
    override_record_name_policy => "true",
    record_name_policy=>"Allow Underscore"
);
unless ($srg) {
   die("Construct DNS SRG failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS SRG object created successfully\n";

#Add the DNS SRG record object to Infoblox Appliance through a session
$session->add($srg)
    or die("Add SRG object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DNS SRG object added to server successfully\n";
```

#Search for a specific DNS SRG record

```
#Search all SRG records with name containing 'test'
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::SRG",
    name   => ".*test.*"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Search SRG failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search DNS SRG object found at least 1 matching entry\n";
```

#Get and modify a DNS SRG record

```
#Get SRG object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SRG",
    name   => "test_srg"
);
my $object = $retrieved_objs[0];
```

```
unless ($object) {
    die("Get SRG failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get DNS SRG object found at least 1 matching entry\n";

#Modify one of the attributes of the specified SRG object
$object->comment("Updated comment.");

#Apply the changes
$session->modify($object)
    or die("Modify SRG object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "DNS SRG object modified successfully \n";
```

#Remove a DNS SRG object

```
#Get SRG object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::SRG",
    name   => "test_srg"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get SRG object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get SRG object found at least 1 matching entry\n";


#Submit the object for removal
$session->remove($object)
    or die("Remove SRG object failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "DNS SRG object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::Zone, Infoblox::Grid::DNS, Infoblox::Grid::RecordNamePolicy, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::TSIGKey - DNS TSIGKey object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DNS->allow_query( )
  - Infoblox::Grid::Member::DNS->allow_query( )
  - Infoblox::DNS::Zone->allow_query( )
  - Infoblox::Grid::DNS->allow_transfer( )
  - Infoblox::Grid::Member::DNS->allow_transfer( )
  - Infoblox::DNS::Zone->allow_transfer( )
  - Infoblox::Grid::DNS->allow_update( )
  - Infoblox::Grid::Member::DNS->allow_update( )
  - Infoblox::DNS::Zone->allow_update( )
  - Infoblox::Grid::DNS->recursive_query_list( )
  - Infoblox::Grid::Member::DNS->recursive_query_list( )
- METHODS
  - key( )
  - name( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::TSIGKey - DNS TSIGKey object

DESCRIPTION

On a device you configure as a secondary serer for a zone, you can associate a TSIG key for each primary server to which the secondary server requests zone transfers. On the device you configure as a primary server for a zone, you can set a TSIG key at the grid, member, or zone level. When the secondary server requests zone transfers, it must send a specific key in its requests to the primary server. When the primary server responds to the requests, it can have a set of TSIG keys from which it can draw when responding. As long as the primary server can find the same TSIG key that the secondary sends it, it can verify the authenticity of the requests it receives and authenticate the responses it sends.

TSIG is based on DNS specifications RFC 2845, TSIG and RFC 2930, TKEY which provides authentication protocol enhancements to secure Server-Server transactions.

CONSTRUCTOR

```
my $tsig_key = Infoblox::DNS::TSIGKey->new(
   key  => $string | ":2xCOMPAT",  #Optional / Default is empty
   name => $string,                #Optional / Default is empty
 );
```

MODULE METHODS

This section describes all the functions that you can apply to a TSIGKey object.

Infoblox::Grid::DNS->allow_query( )

Use this function to specify a TSIG key for a Grid DNS object. This key is used to authenticate clients from which DNS queries are allowed.

Example

```
## Construct the object
my $tsig_key = Infoblox::DNS::TSIGKey->new(
    key  => "Zx1jVJl7C58GT/sc7Q3ucA==",
    name => "test_key"
);
#Adding a TSIG key to the allow_query list of the Grid DNS object
#Retrieving the list
my $allow_query_ref= $Grid_DNS->allow_query();
push @$allow_query_ref, $tsig_key;
#Modifying the object;
```

```
my $response = $Grid_DNS->allow_query($allow_query_ref);
```

Infoblox::Grid::Member::DNS->allow_query( )

Use this function to specify a TSIG key for a Grid Member DNS object. This key is used to authenticate clients from which DNS queries are allowed.

Example

```
##Construct the object
my $tsig_key = Infoblox::DNS::TSIGKey->new(
    key  => "ZX1jVjl7C58GT/sc7Q3ucA==",
    name => "test_key"
);
#Adding a TSIG key to the allow_query list of the Grid Member DNS object
#Retrieving the list
my $allow_query_ref= $Grid_Member_DNS->allow_query();
push @$allow_query_ref, $tsig_key;
#Modifying the object;
my $response = $Grid_Member_DNS->allow_query($allow_query_ref);
```

Infoblox::DNS::Zone->allow_query( )

Use this function to specify a TSIG key for a Zone object. This key is used to authenticate clients from which DNS queries are allowed.

Example

```
## Construct the object
my $tsig_key = Infoblox::DNS::TSIGKey->new(
    key  => "ZX1jVJl7c58GT/sc7Q3ucA==",
    name => "test_key"
);
#Adding a TSIG key to the allow_query list of a zone
#Retrieving the list
my $allow_query_ref= $zone->allow_query();
push @$allow_query_ref, $tsig_key;
#Modifying the object;
my $response = $zone->allow_query($allow_query_ref);
```

Infoblox::Grid::DNS->allow_transfer( )

Use this function to specify a TSIG key for a Grid DNS object. This key is used during zone transfers from the primary server to the secondary servers.

Example

```
## Construct the object
my $tsig_key = Infoblox::DNS::TSIGKey->new(
    key  => "Zx1jVJl7C58GT/sc7Q3ucA==",
    name => "test_key"
);
#Adding a TSIG key to the allow_transfer list of the Grid DNS object
#Retrieving the list
my $allow_transfer_ref= $Grid_DNS->allow_transfer();
push @$allow_transfer_ref, $tsig_key;
#Modifying the object;
my $response = $Grid_DNS->allow_transfer($allow_transfer_ref);
```

Infoblox::Grid::Member::DNS->allow_transfer( )

Use this function to specify a TSIGKey for a grid member object. This key is used during zone transfers from the primary server to the secondary servers.

Example

```
## Construct the object
my $tsig_key = Infoblox::DNS::TSIGKey->new(
    key  => "ZX1jVjl7C58GT/sc7Q3ucA==",
    name => "test_key"
);
#Adding a TSIG key to the allow_transfer list of the Grid Member DNS object
#Retrieving the list
my $allow_transfer_ref= $Grid_Member_DNS->allow_transfer();
push @$allow_transfer_ref, $tsig_key;
#Modifying the object;
my $response = $Grid_Member_DNS->allow_transfer($allow_transfer_ref);
```

Infoblox::DNS::Zone->allow_transfer( )

Use this function to specify a TSIGKey for a Zone object. This key is used during zone transfers from the primary server to the secondary servers.

Example

```
## Construct the object
 my $tsig_key = Infoblox::DNS::TSIGKey->new(
   key  => "ZX1jVJl7c58GT/sc7Q3ucA==",
   name => "test_key"
 );
 #Adding a TSIG key to the allow_transfer list of a zone
 #Retrieving the list
 my $allow_transfer_ref= $zone->allow_transfer();
 push @$allow_transfer_ref, $tsig_key;
 #Modifying the object;
 my $response = $zone->allow_transfer($allow_transfer_ref);
```

Infoblox::Grid::DNS->allow_update( )

Use this function to specify a TSIGKey for a grid DNS object. This key is used during dynamic DNS updates.

Example

```
## Construct the object
 my $tsig_key = Infoblox::DNS::TSIGKey->new(
   key  => "ZX1jVJl7C58gT/sc7Q3ucA==",
   name => "test_key"
 );
 #Adding a TSIG key to the allow_update list of the Grid DNS object
 #Retrieving the list
 my $allow_update_ref= $Grid_DNS->allow_update();
 push @$allow_update_ref, $tsig_key;
 #Modifying the object;
 my $response = $Grid_DNS->allow_update($allow_transfer_ref);
```

Infoblox::Grid::Member::DNS->allow_update( )

Use this function to specify a TSIGKey for a grid member object. This key is used during dynamic DNS updates.

Example

```
## Construct the object
my $tsig_key = Infoblox::DNS::TSIGKey->new(
    key  => "ZX1jVJl7C58Gt/sc7Q3ucA==",
    name => "test_key"
);
# Configure a TSIG key for a Grid Member DNS object
#Retrieving the list
my $allow_update_ref= $Grid_Member_DNS->allow_update();
push @$allow_update_ref, $tsig_key;
#Modifying the object;
my $response = $Grid_Member_DNS->allow_update($allow_transfer_ref);
```

Infoblox::DNS::Zone->allow_update( )

Use this function to specify a TSIGKey for a Zone object. This key is used during dynamic DNS updates.

Example

```
## Construct the object
 my $tsig_key = Infoblox::DNS::TSIGKey->new(
   key  => "ZX1jVJl7C58GT/sc7q3ucA==",
   name => "test_key"
 );
 # Configure TSIGKey on a Zone
 #Retrieving the list
 my $allow_update_ref= $zone->allow_update();
 push @$allow_update_ref, $tsig_key;
 #Modifying the object;
 my $response = $zone->allow_update($allow_transfer_ref);
```

Infoblox::Grid::DNS->recursive_query_list( )

Use this function to specify a TSIG key for a Grid DNS object. This key is used to authenticate clients from which the grid is allowed to receive recursive queries.

Example

```
## Construct the object
my $tsig_key = Infoblox::DNS::TSIGKey->new(
  key  => "ZX1jVJl7C58gT/sc7Q3ucA==",
  name => "test_key"
);
#Adding a TSIG key to the recursive_query_list list of the Grid DNS object
#Retrieving the list
my $recursive_query_list_ref= $Grid_DNS->recursive_query_list();
push @$recursive_query_list_ref, $tsig_key;
#Modifying the object;
my $response = $Grid_DNS->recursive_query_list($allow_transfer_ref);
```

Infoblox::Grid::Member::DNS->recursive_query_list( )

Use this function to specify a TSIG key for a Grid Member DNS object. This key is used to authenticate clients from which recursive queries are allowed.

Example

Example

```
## Construct the object
my $tsig_key = Infoblox::DNS::TSIGKey->new(
    key  => "ZX1jVJl7C58Gt/sc7Q3ucA==",
    name => "test_key"
);

# Configure a TSIG key for a Grid Member DNS object
#Retrieving the list
my $recursive_query_list_ref= $Grid_Member_DNS->recursive_query_list();
push @$recursive_query_list_ref, $tsig_key;
#Modifying the object;
my $response = $Grid_Member_DNS->recursive_query_list($allow_transfer_ref);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a TSIGKey object.

key( )

Use this method to add or retrieve a TSIGKey. To use an existing TSIG key, type or paste the key in the Key field. The TSIG key string is a text string with a key generated by an encryption algorithm

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

For TSIG zone transfers, valid values for a key are `":2xCOMPAT"` or a TSIG key string. `":2xCOMPAT"` means you are using Infoblox DNSone 2.x TSIG. If the key is `":2xCOMPAT"`, the name can only be `"tsig_xfer"`. For TSIG dynamic updates, the valid value a for key is a TSIG key string.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting key
my $key = $tsig_key->key();
# Modifying key
$tsig_key->key("ZX1jVJl7C58GT/sc7Q3ucB==");
```

name( )

Use this method to set or retrieve the name of the TSIGKey.

Enter a meaningful name for the key, such as a zone name or the name of the remote name server with which the local server authenticates zone transfer requests and replies. This name must match the name of the TSIG key on the other name server that use it to authenticate zone transfers with the local server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the TSIG key name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting name
my $name = $tsig_key->name();
# Modifying name
$tsig_key->name("tsig_key1");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add and modify. This sample also includes error handling for the operations.

##Preparation prior to a DNS TSIGKey object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a TSIGKey object

```
 my $tsig_key = Infoblox::DNS::TSIGKey->new(
    key  => "ZX1jVJl7C58GT/sc7Q3ucA==",
    name => "test_key"
  );


unless($tsig_key) {
     die("Construct TSIGKey failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "TSIGKey object created successfully\n";

#Get the member DNS object and add a TSIGKey object to it

my @result_array = $session->get(
    object => "Infoblox::Grid::Member::DNS",
    name   => "infoblox.localdomain"
);

unless (scalar(@result_array) == 0) {
    my $memberdns = $result_array[0];
    if ($memberdns) {

       #Add a TSIG key to the Grid Member DNS object allow_transfer list.
       my $allow_transfer_ref=$memberdns->allow_transfer();
       push @$allow_transfer_ref, $tsig_key;
       $memberdns->allow_transfer($allow_transfer_ref)
            or die("modify member failed: ",
                   Infoblox::status_code() . ":" . Infoblox::status_detail());
        print "TSIGKey added to Member DNS object successfully\n";

    #Update the Grid Member DNS object through the Infoblox session.
        $session->modify($memberdns)
            or die("modify session failed: ",
                   $session->status_code() . ":" . $session->status_detail());
        print "Member DNS object with TSIGKey updated to Infoblox device successfully\n";
```

```
        #Modify existing TSIGKey object
        #Modifying the value of the specified object.

        $tsig_key->name("another_test_key");
        print "Modify TSIG name value\n";

        #Replace the key in the allow_transfer list
        my $allow_transfer_ref2=$memberdns->allow_transfer();

        if(ref($allow_transfer_ref2) eq 'ARRAY'){
            map { if((ref($_) eq 'Infoblox::DNS::TSIGKey')&&($_->name() eq 'test_key')){ $_=$tsig_key;} }
@$allow_transfer_ref2;
        }

        $memberdns->allow_transfer($allow_transfer_ref2)
            or die("modify member failed: ",
                   Infoblox::status_code() . ":" . Infoblox::status_detail());
        print "TSIG name updated to Member DNS object successfully\n";

       #Update the Grid Member DNS object through the Infoblox session.
        $session->modify($memberdns)
            or die("modify session failed: ",
                $session->status_code() . ":" . $session->status_detail());
                print "Member DNS object with TSIG name updated to Infoblox device successfully\n";
             }
         } else {
             print "No member found with the specified name.";
         }

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Grid::Member::DNS, Infoblox::DNS::Zone, Infoblox::Grid::DNS

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::View - DNS View object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - blacklist_action( )
  - blacklist_log_query( )
  - blacklist_redirect_addresses( )
  - blacklist_redirect_ttl( )
  - blacklist_rulesets( )
  - comment( )
  - custom_root_name_servers( )
  - disable( )
  - dns64_groups( )
  - dnssec_enabled( )
  - dnssec_expired_signatures_enabled( )
  - dnssec_validation_enabled( )
  - dnssec_trusted_keys( )
  - enable_blacklist( )
  - enable_dns64( )
  - extensible_attributes( )
  - match_clients( )
  - name( )
  - network_view( )
  - notify_delay( )
  - override_blacklist( )
  - override_dns64( )
  - override_dnssec( )
  - recursion( )
  - use_root_name_servers( )
  - override_nxdomain_redirect( )
  - nxdomain_redirect( )
  - nxdomain_redirect_addresses( )
  - nxdomain_redirect_ttl( )
  - nxdomain_log_query( )
  - nxdomain_rulesets( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::View - DNS View object

DESCRIPTION

DNS views provide the ability to serve one version of DNS data to one set of clients and another version to another set of clients. With DNS views, the Infoblox appliance can provide a different answer to the same DNS query, depending on the source of the query. When the DNS view is not specified, all objects are automatically associated with the default DNS view.

CONSTRUCTOR

```
 my $view = Infoblox::DNS::View->new(
     name                     => $string,
#Required
     blacklist_action         => "REDIRECT" | "REFUSE",
#Optional / Default is "REFUSE"
     blacklist_log_query      => "true" | "false",
#Optional / Default is "false"
     blacklist_redirect_addresses => [$ipv4addr, ... ],
#Optional / Default is undefined
     blacklist_redirect_ttl   => $num,
```

```
#Optional / Default is 60
     blacklist_rulesets       => [ $rulesetname1, $rulesetname2, ...],
#Optional / Default is undefined
     comment                  => $string,
#Optional / Default is empty
     custom_root_name_servers => [$RootNameServer1, $RootNameServer2, ...],
#Optional / Default is undefined
     disable                  => "true" | "false",
#Optional / Default "false"
     dns64_groups             => [$group1, $group2, ... ],
#Optional / Default is undefined
     dnssec_enabled           => "true" | "false" | undef,
#Optional / Default is undefined
     dnssec_expired_signatures_enabled => "true" | "false",
#Optional / Default is "false"
     dnssec_validation_enabled         => "true" | "false",
#Optional / Default is "true"
     dnssec_trusted_keys               => [$key1, $key2, ...]
#Optional / Default is undefined
     enable_blacklist         => "true" | "false" | undef,
#Optional / Default is undefined
     enable_dns64             => "true" | "false" | undef,
#Optional / Default is undefined
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     match_clients            => [$ipv4addr | $ipv4_subnet/prefix | $ipv6addr |
                                 $ipv6_subnet/prefix | "any" | $TSIGKey ...],
#Optional / Default is undefined
     network_view             => $Object,
#Optional / Default is "default" network view
     notify_delay             => $num,
#Optional / Default is undefined
     override_dnssec          => "true"| "false",
#Optional / Default is "false"
     override_blacklist       => "true" | "false",
#Optional / Default is "false"
     override_dns64           => "true" | "false",
#Optional / Default is "false"
     override_nxdomain_redirect => "true"| "false",
#Optional / Default is "false"
     recursion                => "true" | "false",
#Optional / Default "false"
     use_root_name_servers    => "true" | "false",
#Optional / Default "false"
     nxdomain_redirect        => "true" | "false",
#Optional / Default is "false"
     nxdomain_redirect_addresses => [$ipv4addr, ...],
#Optional / Default is undefined
     nxdomain_redirect_ttl    => $num
#Optional / Default is 60
     nxdomain_log_query       => "true" | "false",
#Optional / Default is "false"
     nxdomain_rulesets        => [$reulesetname1, $reulesetname2, ...]
#Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a DNS view object.

Infoblox::Session->add( )

Use this method to add a DNS view object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#create member
my $member1 = Infoblox::DNS::Member->new(
    name     => "member.com",
    ipv4addr => "192.168.1.2",
    lead     => "false",
    stealth  => "false"
);

#create keys
my $key1 = Infoblox::DNS::TSIGKey->new(
    name => "tsig_xfer",
    key  => ":2xCOMPAT"
);
```

```
my $tsig_key2 = $session->gen_tsig_key();

my $key2 = Infoblox::DNS::TSIGKey->new(
    name => "key2",
    key  => $tsig_key2
);

#create view object using members
my $firstview = Infoblox::DNS::View->new(
    name              => "internal",
    match_clients     => ["1.1.1.1", $key1 , "10.0.0.0/8", $key2],
    recursion         => "true",
    disable           => "false",
    comment           => "new Internal view"
    )
    or die(qq(constructor for view internal failed: ),
    join(":", Infoblox::status_code(), Infoblox::status_detail()));

#Add a  View through session
$session->add($firstview);
unless ($firstview) {
         die("add view internal failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
```

Infoblox::Session->get( )

Use this method to retrieve an existing DNS view object from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

Apply the following attributes to get a specific DNS view object:

```
name - Required. A name in string format.
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
#get view object from session
my @result = $session->get(
    object => "Infoblox::DNS::View",
    name   => "internal"
);

my $firstview = $result[0];

# get all DNS view objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::View",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify a DNS view object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment for view object.
$firstview->comment("new comment internal view");
# Submit modification
my $response = $session->modify($firstview);
```

Infoblox::Session->remove( )

Use this function to submit to remove a DNS view object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

#Remove an object with a unique name

```
my @result = $session->search(object => "Infoblox::DNS::View", name => "^internal\$");
my $firstview = $result[0];
```

```
#Submit for removal
my $response = $session->remove($firstview);
```

Infoblox::Session->search( )

Use this method to search for an existing DNS view object in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

Apply the following attributes to search for a DNS view object:

```
name    - Required. A name in string format (regular expression).
extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
#Search for a DNS view by name
my @result = $session->search(object => "Infoblox::DNS::View", name => "^internal\$");

# search for DNS views with the extensible attribute 'Site'
my @retrieved_objs = $session->search(
   object => "Infoblox::DNS::View",
   extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute value of a DNS view object.

blacklist_action( )

Use this method to specify the action to perform when a domain name matches the pattern defined in a rule that is specified by the blacklist_ruleset method.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "REDIRECT" or "REFUSE". The default value is "REFUSE".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Examples

```
#Get blacklist_action
my $blacklist_action = $firstview->blacklist_action();
#Modify blacklist_action
$firstview->blacklist_action("REDIRECT");
```

blacklist_log_query( )

Use this method to set or retrieve the flag that indicates whether blacklist redirection queries are logged.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable logging, or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_log_query
my $blacklist_log_query = $firstview->nxdomain_log_query( );
#Modify blacklist_log_query
$firstview->blacklist_log_query("true");
```

blacklist_redirect_addresses( )

Use this method to specify the IP addresses the appliance includes in the response it sends in place of a blacklisted IP address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array of IPv4 addresses. The default value is undefined.

Setting this parameter to ["10.0.0.10", "10.0.0.20"] or [], implicitly sets the override_blacklist_redirect attribute to "true". Setting this parameter to undefined causes the appliance to use the member-level blacklist redirection settings and automatically sets the override_blacklist_redirect attribute to "false" if "enable_blacklist_redirect" is also undefined. But if "enable_blacklist_redirect" contains a value, then the override_blacklist_redirect attribute is set to "true".

Note that when blacklist_redirect_addresses contains a valid array reference and override_blacklist_redirect is set to "false", the last operation takes precedence. Thus the sequence $firstview->blacklist_redirect_addresses(["10.0.0.10", "10.0.0.20"]); $firstview->override_blacklist_redirect("false"); sets override_blacklist_redirect to "false", and the sequence $firstview->override_blacklist_redirect("false"); $firstview->blacklist_redirect_addresses(["10.0.0.10", "10.0.0.20"]); sets override_blacklist_redirect to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_redirect_addresses
my $blacklist_redirect_addresses = $firstview->blacklist_redirect_addresses( );
#Modify blacklist_redirect_addresses
$firstview->blacklist_redirect_addresses(["10.0.0.10", "10.0.0.20"]);
```

blacklist_redirect_ttl( )

Use this method to set or retrieve the TTL value of the synthetic DNS responses that result from blacklist redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer, from 0 to 4294967295, that represents the TTL in seconds. The default value is 60.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_redirect_ttl
my $blacklist_redirect_ttl = $firstview->blacklist_redirect_ttl( );
#Modify blacklist_redirect_ttl
$firstview->blacklist_redirect_ttl(60);
```

blacklist_rulesets( )

Use this method to specify or retrieve Infoblox::DNS::Ruleset object names assigned at the grid level for blacklist redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains the names of the Infoblox::DNS::Ruleset objects assigned to the grid. The names must be in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_rulesets
my $blacklist_rulesets = $firstview->blacklist_rulesets();
#Modify blacklist_rulesets
$firstview->blacklist_rulesets(["ruleset1", "ruleset2"]);
```

comment( )

Use this method to set or retrieve the comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get comment
my $comment = $firstview->comment();
#Modify comment
$firstview->comment("modifying the internal view comment");
```

custom_root_name_servers( )

Use this method to set or retrieve the list of customized root name servers. You can either select and use Internet root name servers or specify custom root name servers by providing a host name and IP address to which the Infoblox appliance can send queries.Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::DNS::RootNameServer object(s).

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get custom_root_name_servers
my $rootNS_list = $firstview->custom_root_name_servers();
#Modify the root nameserver list
#Creating RootNameServer objects.
my $rootns1 = Infoblox::DNS::RootNameServer->new(
    ipv4addr  => "4.4.4.4",
    host_name => "rns1.test.com",
    );

my $rootns2 = Infoblox::DNS::RootNameServer->new(
    ipv4addr  => "5.5.5.5",
    host_name => "rns2.test.com",
    );

$firstview->custom_root_name_servers([$rootns1, $rootns2]);
```

disable( )

Use this method to set or retrieve the disable flag of a DNS view object.

Include the specific parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false; therefore, the DNS view is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get disable
my $disable = $firstview->disable();
#Modify disable
$firstview->disable("true");
```

dns64_groups( )

Use this method to set or retrieve the list of DNS64 synthesis groups associated with this DNS view.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_dns64 method to "true". Setting the parameter to undefined causes the appliance to use the grid DNS default and automatically resets the override_dns64 method to "false".

Note that when dns64_groups is set to a defined value and override_dns64_groups is set to "false", the last operation takes precedence. Thus the sequence $object->dns64_groups([$group1]); $object->override_dns64("false"); will set override_dns64 to "false", and the sequence $object->override_dns64([$group1]); $object->dns64_groups("testdomain"); will result in override_dns64="true".

Parameter

Valid value is an array reference that contains Infoblox::Grid::DNS::DNS64Group objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the list of DNS64 synthesis groups
my $dns64_groups_list = $firstview->dns64_groups();

#Modify it
my $group = $session->get(
                        object => "Infoblox::Grid::DNS::DNS64Group"),
                        name => "groupname",
                        prefix => "groupprefix",
                      );
$firstview->dns64_groups([$group]);
```

dnssec_enabled( )

Use this method to enable/disable the DNS security extension.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to enable the DNS security extension. Set it to "false" to disable the feature.

Setting this parameter to "true"/"false" implicitly sets the override_dnssec attribute to "true". Setting the parameter to undefined causes the appliance to use the member-level DNS security settings and automatically resets the override_dnssec attribute to "false".

Note that in the case when dnssec_enabled is set to "true"/"false" and override_dnssec is set to "false" simultaneously, the last operation takes precedence. Thus the sequence $firstview->dnssec_enabled("false"); $firstview->override_dnssec("false"); will set override_dnssec to "false", and the sequence $firstview->override_dnssec("false"); $firstview->dnssec_enabled("false"); will result in override_dnssec="true".

The default value of the parameter is undefined, which means the DNS view inherits the member-level settings.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_enabled
my $dnssec_enabled = $firstview->dnssec_enabled();
#Modify dnssec_enabled
$firstview->dnssec_enabled("true");
#Set the value to undefined to inherit the member-level settings.
$firstview->dnssec_enabled(undef);
```

dnssec_expired_signatures_enabled( )

Use this method to control whether DNS accepts expired signatures.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to accept expired signatures. Set it to "false" to reject expired signatures.

The default value of the parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_expired_signatures_enabled
my $dnssec_expired_signatures_enabled = $firstview->dnssec_expired_signatures_enabled();
#Modify dnssec_expired_signatures_enabled
$firstview->dnssec_expired_signatures_enabled("true");
```

dnssec_validation_enabled( )

Use this method to enable/disable DNS security validation.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to enable the validation. Set the parameter to "false" to disable the validation.

The default value of the parameter is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_validation_enabled
my $dnssec_validation_enabled = $firstview->dnssec_validation_enabled();
#Modify dnssec_validation_enabled
$firstview->dnssec_validation_enabled("false");
```

dnssec_trusted_keys( )

Use this method to specify the list of trusted keys for the DNS security extension.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference that contains a list of Infoblox::DNS::DnssecTrustedKey objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns a reference to a list of the trusted keys.

Example

```
#Get dnssec_trusted_keys
my $dnssec_trusted_keys = $firstview->dnssec_trusted_keys();
#Modify dnssec_trusted_keys
my $key1=Infoblox::DNS::DnssecTrustedKey->new(
   fqdn => "myzone.com",
   algorithm => "RSAMD5",
   key       => "c2ltcGxlIGtleQo="
   );
$firstview->dnssec_trusted_keys([$key1]);
```

enable_blacklist( )

Use this method to enable or disable a blacklist in a DNS view.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_blacklist flag or "false" to deactivate/unset it. The default is "false".

Setting this parameter to "true" or "false" implicitly sets the override_blacklist attribute to "true". Setting the parameter to undefined causes the appliance to use the member-level blacklist settings and automatically sets the override_blacklist attribute to "false" if "blacklist_redirect_addresses" is also undefined.

Note that when enable_blacklist is set to "true"/"false" and override_blacklist is set to "false", the last operation takes precedence. Thus the sequence $firstview->enable_blacklist("false"); $firstview->override_blacklist("false"); sets override_blacklist to "false", and the sequence $firstview->override_blacklist("false"); $firstview->enable_blacklist("false"); sets override_blacklist to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_blacklist
my $enable_blacklist = $firstview->enable_blacklist( );
#Modifying enable_blacklist
$firstview->enable_blacklist("true");
```

enable_dns64( )

Use this method to enable/disable DNS64 support for this view.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_dns64 method to "true". Setting the parameter to undefined causes the appliance to use the grid DNS default and automatically resets the override_dns64 method to "false".

Note that when enable_dns64 is set to a defined value and override_enable_dns64 is set to "false", the last operation takes precedence. Thus the sequence $object->enable_dns64("true"); $object->override_dns64("false"); will set override_dns64 to "false", and the sequence $object->override_dns64("true"); $object->enable_dns64("testdomain"); will result in override_dns64="true".

Parameter

Specify "true" to set the enable_dns64 flag or "false" to deactivate and unset it. The default value is "false".

Returns

When you specify a parameter and the modification succeeds, the method returns "true". When the operation fails, the method returns "false".

When you do not specify a parameter, the method returns the attribute value.

Example

```
#Get the enable_dns64 flag
my $enable_dns64 = $firstview->enable_dns64();
#Modify the enable_dns64 flag
$firstview->enable_dns64("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS view object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $firstview->extensible_attributes();
#Modify extensible attributes
$firstview->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
```

```
});
```

match_clients( )

Use this method to set or retrieve the match_clients list. This list specifies IP addresses, networks and/or TSIG keys of the clients that are allowed or denied access to the DNS view.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 or IPv6 addresses or networks. To deny access to IP addresses and/or networks, add "!" as a prefix to IP addresses and/or networks. Specify "any" to allow access to any client. To allow access to a client authorized by a TSIG key, use Infoblox::DNS::TSIGKey objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get match_clients
my $clients = $firstview->match_clients();
#Modify match_clients
$firstview->match_clients(["any"]);

#Specifying TSIG keys
my $tsig_key3 = $session->gen_tsig_key();
my $key3 = Infoblox::DNS::TSIGKey->new(
    name => "key3",
    key  => $tsig_key3
);
#Modify match_clients
$firstview->match_clients(["10.0.0.10" , $key3, "20.0.0.0/24"
,"2001::123a","2003::222/64","!"."30.0.0.30" ,
"!"."40.10.0.0/24","!"."2004::125b","!"."2006::34bc"]);
```

name( )

Use this method to configure or retrieve the name of a DNS view.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the DNS view in string format. It has a maximum of 64 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get name
my $name = $firstview->name();
#Modify name
$firstview->name("external");
```

network_view( )

Use this method to set or retrieve the network view object associated with this DNS view object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The value should be a reference of defined Infoblox::DHCP::View objects.

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you do not specify a parameter, the method returns the attribute value.

Example

```
#Get network_view
my $useRootNS = $firstview->network_view( );
#Modify network_view
```

```
$firstview->use_root_name_servers($network_view);
```

notify_delay( )

Use this method to specify with how many seconds of delay the notify messages are sent to secondaries.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are unsigned integer between 5 and 86400, inclusive. Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting notify_delay
my $notify_delay = $firstview->notify_delay( );
#Modifying notify_delay
$firstview->notify_delay(5);
```

override_blacklist( )

The "override_blacklist" attribute controls whether the "enable_blacklist", "blacklist_redirect_addresses", "blacklist_redirect_ttl", "blacklist_log_query", and "blacklist_rulesets" values in the database are used.

The "override_blacklist" attribute can be specified explicitly. It is also set implicitly as follows: "enable_blacklist" => "true"|"false" or "blacklist_redirect_addresses" => ["10.0.0.10", "10.0.0.20"]|[] results in "override_blacklist" => "true", "enable_blacklist" => undef and "blacklist_redirect_addresses" => undef results in "override_blacklist" => "false". If "override_blacklist" is specified explicitly as "true", and "enable_blacklist" => undef or "blacklist_redirect_addresses" => undef, then the explicit value takes precedence. Therefore, "override_blacklist"=>"true", "enable_blacklist"=>undef, and "blacklist_redirect_addresses"=>undef remain unchanged in the database.

Specifying the "blacklist_redirect_ttl", "blacklist_log_query", or "blacklist_rulesets" attributes does not affect the "override_blacklist" value. It is implicitly set only by "enable_blacklist" and "blacklist_redirect_addresses".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the member-level settings of the blacklist redirect attributes. Set the parameter to "false" to inherit the member-level settings. The default is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_blacklist
my $override_blacklist=$firstview->override_blacklist( );
#Modify override_blacklist
$firstview->override_blacklist("true");
```

override_dns64( )

The override_dns64 attribute controls whether the DNS64 method values in this DNS view are used, instead of the grid default.

The override_dns64 attribute can be specified explicitly. It is also set implicitly when enable_dns64 or dns64_groups are set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for enable_dns64 and dns64_groups. Set the parameter to "false" to inherit the grid-level setting for enable_dns64 and dns64_groups.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_dns64
my $override_dns64=$firstview->override_dns64( );
#Modifying override_dns64
$firstview->override_dns64("true");
```

override_dnssec( )

The override_dnssec attribute controls whether the dnssec_enabled, dnssec_validation_enabled, dnssec_expired_signatures_enabled, and dnssec_trusted_keys values in the database are used.

The override_dnssec attribute can be specified explicitly. It is also established implicitly as follows: "dnssec_enabled" => "true"|"false" results in "override_dnssec" => "true" "dnssec_enabled" => undef results in "override_dnssec" => "false" If the override_dnssec attribute is specified explicitly as "true", while dnssec_enabled=>undef, the explicit value takes precedence, and override_dnssec=>"true" while dnssec_enabled remains unchanged in the DB.

Supplying the dnssec_validation_enabled, dnssec_expired_signatures_enabled, or dnssec_trusted_keys attributes does not affect the override_dnssec value. It is implicitly affected only by dnssec_enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the member-level setting of the DNSSEC attributes. Set the parameter to "false" to inherit the grid-member level settings of DNSSec.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_dnssec
my $override_dnssec=$firstview->override_dnssec( );
#Modifying override_dnssec
$firstview->override_dnssec("true");
```

recursion( )

Use this method to enable/disable recursion.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the recursion flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get recursion
my $recursion = $firstview->recursion();
#Modify recursion
$firstview->recursion("false");
```

use_root_name_servers( )

Use this method to enable/disable custom or Internet root name servers. This setting overrides grid root name server settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable custom root name servers. Specify "false" to use Internet root name servers.

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you do not specify a parameter, the method returns the attribute value.

Example

```
#Get use_root_name_servers
my $useRootNS = $firstview->use_root_name_servers( );
#Modify use_root_name_servers
$firstview->use_root_name_servers("false");
```

override_nxdomain_redirect( )

The "override_nxdomain_redirect" attribute controls whether the "nxdomain_redirect", "nxdomain_redirect_addresses", "nxdomain_redirect_ttl", "nxdomain_log_query", and "nxdomain_rulesets" values for this DNS view are used.

The "override_nxdomain_redirect" attribute can be specified explicitly. It is also set implicitly as follows: "nxdomain_redirect" => "true"|"false" or "nxdomain_redirect_addresses" => ["10.0.0.10", "10.0.0.20"]|[] result in "override_nxdomain_redirect" => "true", "nxdomain_redirect" => undef and "nxdomain_redirect_addresses" => undef result in "override_nxdomain_redirect" => "false". If "override_nxdomain_redirect" is specified explicitly as "true", and "nxdomain_redirect" => undef or "nxdomain_redirect_addresses" => undef, then the explicit value takes precedence. Therefore, "override_nxdomain_redirect"=>"true", "nxdomain_redirect"=>undef, and "nxdomain_redirect_addresses"=>undef remain unchanged in the database.

Specifying the "nxdomain_redirect_ttl", "nxdomain_log_query", or "nxdomain_rulesets" attributes does not affect the "override_nxdomain_redirect" value. It is implicitly set only by "nxdomain_redirect".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the member-level settings of the NXDOMAIN redirect attributes. Set the parameter to "false" to inherit the member-level settings. The default is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_nxdomain_redirect
my $override_nxdomain_redirect=$firstview->override_nxdomain_redirect( );
#Modify override_nxdomain_redirect
$firstview->override_nxdomain_redirect("true");
```

nxdomain_redirect( )

Use this method to enable or disable NXDOMAIN redirection in a DNS view.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the nxdomain_redirect flag or "false" to deactivate/unset it. The default is "false".

Setting this parameter to "true" or "false" implicitly sets the override_nxdomain_redirect attribute to "true". Setting the parameter to undefined causes the appliance to use the member-level NXDOMAIN redirection settings and automatically sets the override_nxdomain_redirect attribute to "false" if nxdomain_redirect_addresses is also undefined. But if nxdomain_redirect_addresses contains a value, then the override_nxdomain_redirect attribute is set to true.

Note that when nxdomain_redirect is set to "true"/"false" and override_nxdomain_redirect is set to "false", the last operation takes precedence. Thus the sequence $firstview->nxdomain_redirect("false"); $firstview->override_nxdomain_redirect("false"); sets override_nxdomain_redirect to "false", and the sequence $firstview->override_nxdomain_redirect("false"); $firstview->nxdomain_redirect("false"); sets override_nxdomain_redirect to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting nxdomain_redirect
my $nxdomain_redirect = $firstview->nxdomain_redirect( );
#Modifying nxdomain_redirect
$firstview->nxdomain_redirect("true");
```

nxdomain_redirect_addresses( )

Use this method to specify the IP addresses the appliance includes in the response it sends in place of an NXDOMAIN response.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array of IPv4 addresses. The default value is undefined.

Setting this parameter to ["10.0.0.10", "10.0.0.20"] or [] implicitly sets the override_nxdomain_redirect attribute to "true". Setting the parameter to undefined causes the appliance to use the member-level NXDOMAIN redirection settings and automatically sets the override_nxdomain_redirect attribute to "false" if nxdomain_redirect is also undefined. But if nxdomain_redirect contains a value, then the override_nxdomain_redirect attribute is set to true.

Note that when nxdomain_redirect_addresses contains a valid array reference and override_nxdomain_redirect is set to "false", the last operation takes precedence. Thus the sequence $firstview->nxdomain_redirect_addresses(["10.0.0.10", "10.0.0.20"]); $firstview->override_nxdomain_redirect("false"); sets override_nxdomain_redirect to "false", and the sequence $firstview->override_nxdomain_redirect("false"); $firstview->nxdomain_redirect_addresses(["10.0.0.10", "10.0.0.20"]); sets override_nxdomain_redirect to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_redirect_addresses
my $nxdomain_redirect_addresses = $firstview->nxdomain_redirect_addresses( );
#Modify nxdomain_redirect_addresses
$firstview->nxdomain_redirect_addresses(["10.0.0.10", "10.0.0.20"]);
```

nxdomain_redirect_ttl( )

Use this method to set or retrieve the TTL value of the synthetic DNS responses that result from NXDOMAIN redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer, from 0 to 4294967295, that represents the TTL in seconds. The default value is 60.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_redirect_ttl
my $nxdomain_redirect_ttl = $firstview->nxdomain_redirect_ttl( );
#Modify nxdomain_redirect_ttl
$firstview->nxdomain_redirect_ttl(60);
```

nxdomain_log_query( )

Use this method to set or retrieve the flag that indicates whether NXDOMAIN redirection queries are logged.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable logging, or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_log_query
my $nxdomain_log_query = $firstview->nxdomain_log_query( );
#Modify nxdomain_log_query
$firstview->nxdomain_log_query("true");
```

nxdomain_rulesets( )

Use this method to specify or retrieve Infoblox::DNS::Ruleset object names assigned at the grid level for NXDOMAIN redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains the names of the Infoblox::DNS::Ruleset objects assigned to the grid. The names must be in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_rulesets
my $nxdomain_rulesets = $firstview->nxdomain_rulesets();
#Modify nxdomain_rulesets
$firstview->nxdomain_rulesets(["ruleset1", "ruleset2"]);
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object, such as create, search, modify, and remove. This sample code also includes error handling for the operations.

#Preparation prior to a DNS view insertion

```
 use strict;
 use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DNS view

```
#create member
my $member1 = Infoblox::DNS::Member->new(
    name     => "member.com",
    ipv4addr => "192.168.1.2",
    lead     => "false",
    stealth  => "false"
);

#create keys
my $key1 = Infoblox::DNS::TSIGKey->new(
    name => "tsig_xfer",
    key  => ":2xCOMPAT"
);

my $tsig_key2 = $session->gen_tsig_key();

my $key2 = Infoblox::DNS::TSIGKey->new(
    name => "key2",
    key  => $tsig_key2
);

#create View object using members
my $firstview = Infoblox::DNS::View->new(
    name              => "internal",
    match_clients     => ["1.1.1.1", $key1, "10.0.0.0/8", $key2],
    recursion         => "true",
    disable           => "false",
    comment           => "new Internal view"
```

```
    );
 unless ($firstview) {
          die("constructor for view internal failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
     }

#put View into session
$session->add($firstview);
unless ($firstview) {
      die("add view internal failed: ",
      $session->status_code() . ":" . $session->status_detail());
}
```

#Search for a specific view

```
#Searching the internal view
my @result = $session->search(object => "Infoblox::DNS::View", name => "^internal\$");
my $object = $result[0];
unless ($object) {
          die("Search view failed: ",
               $session->status_code() . ":" . $session->status_detail());
     }
print "Search view object found at least 1 matching entry\n";
```

#Get and modify a DNS view

```
my @result = $session->get(
    object => "Infoblox::DNS::View",
    name   => "internal"
);

my $firstview = $result[0];
unless ($firstview) {
         die("Get view failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print "Get view found at least 1 matching entry\n";
$firstview->comment("new comment internal view");

#Apply the changes
$session->modify($firstview);
unless ($firstview) {
          die("Modify view failed: ",
               $session->status_code() . ":" . $session->status_detail());
     }
print "View object modified successfully \n";
```

#Remove a view

```
my @result = $session->get(
    object => "Infoblox::DNS::View",
    name   => "internal"
);

my $firstview = $result[0];
unless ($firstview) {
         die("Get view failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print "Get view found at least 1 matching entry\n";

#submit the object for removal
$session->remove($firstview);
unless ($firstview) {
         die("Remove view failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print "View object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DHCP::View, Infoblox::DNS::Zone, Infoblox::DNS::DnssecTrustedKey, Infoblox::DNS::Ruleset, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::DNS::Zone - DNS Zone object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - ad_servers( )
  - allow_gss_tsig_for_underscore_zone( )
  - allow_gss_tsig_zone_updates( )
  - allow_query( )
  - allow_transfer( )
  - allow_update( )
  - allow_update_forwarding( )
  - bind_check_names_policy( )
  - comment( )
  - create_underscore_zones( )
  - delegate_to( )
  - delegated_ttl( )
  - disable( )
  - disable_forwarding( )
  - dnssec_ksk_algorithm( )
  - dnssec_ksk_rollover_interval( )
  - dnssec_ksk_size( )
  - dnssec_signature_expiration( )
  - dnssec_signed( )
  - dnssec_zsk_algorithm( )
  - dnssec_zsk_rollover_interval( )
  - dnssec_zsk_size( )
  - do_host_abstraction( )
  - enable_ad_servers( )
  - email( )
  - extensible_attributes( )
  - forward_only( )
  - forward_to( )
  - host_name_restriction_policy( )
  - import_from( )
  - initiate_ksk_rollover( )
  - locked( )
  - members( )
  - name( )
  - network_associations( )
  - notify_delay( )
  - ms_ad_integrated( )
  - ms_managed( )
  - ms_read_only( )
  - ms_ddns_mode( )
  - ms_allow_transfer( )
  - ns_group( )
  - override_delegated_ttl( )
  - override_grid_zone_timer( )
  - override_grid_email( )
  - override_serial_number( )
  - override_soa_mname( )
  - prefix( )
  - primary( )
  - remove_sub_zones( )
  - secondaries( )
  - soa_default_ttl( )
  - soa_expire( )
  - soa_mname( )
  - soa_negative_ttl( )
  - soa_refresh( )
  - soa_retry( )
  - soa_serial_number( )
  - shared_record_groups( )
  - stub_from( )

- stub_members( )
- views( )
- zone( )

- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::DNS::Zone - DNS Zone object

DESCRIPTION

A zone is a portion of the domain name space for which an Infoblox appliance or another name server is authoritative (for example, has the start of authority [SOA] record). A zone can also be delegated or forwarded. Zones are the primary objects used to manage DNS data and DNS services.

Note that you must specify only one view for the attribute "views".

CONSTRUCTOR

```
 my $zone = Infoblox::DNS::Zone->new(
     name                        => $fqdn | $ipv4_subnet/prefix | $ipv6_subnet/prefix,     #Required
     ad_servers              => [ $ipv4addr | $ipv6addr ],                              #Optional / Default
is empty list
     allow_gss_tsig_for_underscore_zone => "true" | "false",                            #Optional /
Default is "false"
     allow_gss_tsig_zone_updates  => "true" | "false",                                  #Optional /
Default is "false"
     allow_query                 => [ $ipv4addr | $ipv4_subnet/prefix | $ipv6addr |
                                      $ipv6_subnet/prefix | $TSIGKey ,... ] | undef,       #Optional /
Default is undefined
     allow_transfer              => [ $ipv4addr | $ipv4_subnet/prefix | $ipv6addr |
                                      $ipv6_subnet/prefix | $TSIGKey ,... ] | undef,       #Optional /
Default is undefined
     allow_update                => [ $ipv4addr | $ipv4_subnet/prefix | $ipv6addr |
                                      $ipv6_subnet/prefix | $TSIGKey ,... ] | undef,       #Optional /
Default is undefined
     allow_update_forwarding      => [ $ipv4addr | $ipv4_subnet/prefix | $ipv6addr |
                                      $ipv6_subnet/prefix | $TSIGKey ,... ] | undef,        #Optional /
Default is undefined
     bind_check_names_policy      => [ "fail" | "warn" | undef],                             #Optional /
Default is undefined
     comment                     => $string,                                                #Optional /
Default is undefined
     create_underscore_zones      => "true" | "false",                                       #Optional /
Default is "false"
     delegate_to                 => [ $Nameserver1, $Nameserver2 ... ],                     #Optional /
Default is undefined
     delegated_ttl               => $num,                                                   #Optional /
Default is undefined
     disable                     => "true" | "false",                                       #Optional /
Default is "false"
     disable_forwarding           => "true" | "false",                                      #Optional /
Default is "false"
     dnssec_ksk_algorithm         => "RSAMD5"|"DSA"|"RSASHA1"|"NSEC3DSA"|
                                     "NSEC3RSASHA1"|"RSASHA256"|"RSASHA512"|
                                     "NSEC3RSASHA256"|"NSEC3RSASHA512"|1|3|5|6|7            #Optional /
Default is undefined
     dnssec_ksk_rollover_interval => $num,                                                  #Optional /
Default is undefined
     dnssec_ksk_size             => $num,                                                   #Optional /
Default is undefined
     dnssec_signature_expiration  => $num,                                                  #Optional /
Default is undefined
     dnssec_signed               => "true" | "false",                                       #Optional /
Default is undefined
     dnssec_zsk_algorithm         => "RSAMD5"|"DSA"|"RSASHA1"|"NSEC3DSA"|
                                     "NSEC3RSASHA1"|"RSASHA256"|"RSASHA512"|
                                     "NSEC3RSASHA256"|"NSEC3RSASHA512"|1|3|5|6|7            #Optional /
Default is undefined
     dnssec_zsk_rollover_interval => $num,                                                  #Optional /
Default is undefined
     dnssec_zsk_size             => $num,                                                   #Optional /
Default is undefined
     do_host_abstraction          => "true" | "false",                                      #Optional /
```

```
Default is "false"
     enable_ad_servers     => "true" | "false",                                    #Optional / Default is
"false"
     email                 => $email,                                              #Optional /
Default is grid level email setting
     extensible_attributes => { $string => $string | $num, $string => [ $string | $num, ... ], ...
} #Optional / Default is undefined
     forward_only          => "true" | "false",                                    #Optional /
Default is "false"
     forward_to            => [ $Member1, $Member2, $Nameserver1, $Nameserver2,...], #Optional /
Default is undefined
     host_name_restriction_policy => $string,                                      #Optional /
Default is undefined
     import_from           => $ipv4addr,                                           #Optional /
Default is undefined
     locked                => "true" | "false",                                    #Optional /
Default is "false"
     members               => [ $Member1, $Member2,...],                           #Optional /
Default is undefined
     ms_ad_integrated      => "true" | "false"                                     #Optional /
Default is "false"
     ms_allow_transfer     => "Any" | "NSOnly" | "None" | [ $ipv4addr1, ... ]      #Optional /
Default is "NSOnly"
     ms_ddns_mode          => "Secure" | "None" | "Any"                            #Optional /
Default is undefined
     notify_delay          => $num,                                                #Optional /
Default is undefined
     ns_group              => $string,                                             #Optional /
Default is undefined
     override_delegated_ttl => "true" | "false",                                   #Optional /
Default is "false"
     override_grid_zone_timer => "true" | "false",                                 #Optional /
Default is "false"
     override_grid_email   => "true" | "false",                                    #Optional /
Default is "false"
     override_serial_number => "true" | undef,                                     #Optional /
Default is undef
     override_soa_mname    => "true" | "false",                                    #Optional /
Default is "false"
     prefix                => $string,                                             #Optional /
Default is undefined
     primary               => $Member | [ $Nameserver1, $MSServer2, ...] | undef,  #Optional /
Default is undefined
     remove_sub_zones      => "true" | "false",                                    #Optional /
Default is undefined
     secondaries           => [ $Member1, $Member2, $Nameserver1, $MSServer2, ...], #Optional /
Default is empty
     soa_default_ttl       => $num,                                                #Optional /
Default is undefined
     soa_expire            => $num,                                                #Optional /
Default is undefined
     soa_mname             => $fqdn,                                               #Optional /
Default is undefined
     soa_negative_ttl      => $num,                                                #Optional /
Default is undefined
     soa_retry             => $num,                                                #Optional /
Default is undefined
     soa_refresh           => $num,                                                #Optional /
Default is undefined
     soa_serial_number     => $num,                                                #Optional /
Default is 1
     shared_record_groups  => [ $shared_record_group1_name, ... ],                #Optional /
Default is undefined
     stub_from             => [ $Nameserver1, $Nameserver2, ... ],                 #Optional /
Default is undefined
     stub_members          => [ $Member1, $Member2, $MSServer1, $MSServer2, ...],  #Optional /
Default is undefined
     views                 => [$View1],                                            #Optional /
Default is "default" view
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a DNS Zone object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $zone = Infoblox::DNS::Zone->new (
    name     => "domain.com",
    comment  => "this is a demo zone",
    disable  => "false",
    views    => [$view1]
    );
#Submit for addition
my $response = $session->add( $zone );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS Zone object:

 name - Required. A zone name in string format.
 view - Optional. The DNS view in which the zone is located. The default is all DNS views.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Zone",
    name   => "domain.com",
    view   => "default" );

my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Zone",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the comment.
$zone->comment("this is a modified comment");
#Submit modification
my $response = $session->modify( $zone );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Zone",
    name   => "domain.com",
    view   => "default" );
#Find the desired object from the retrieved list.
my $desired_zone = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_zone );
```

Infoblox::Session->search( )

Use this method to search for DNS Zone objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS Zone object:

 name - Required. A host name in string format (regular expression).
 view - Optional. The DNS view in which the zone is located. The default is all DNS views.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 comment - Optional. A comment in string format (regular expression).
 primary_association_type - Optional. "Microsoft" is the only allowed value. If it is specified, the
search returns only zones with primary servers that are Microsoft (r) DNS servers.
```

Examples

```
# search for all DNS zone objects under com in default view
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Zone",
    name   => ".*\\.com",
    view   => "default" );

# search for all DNS zone objects defining "Santa Clara" for "Site" extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Zone",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a zone.

ad_servers( )

Use this method to set or retrieve the list of AD domain controllers that are allowed to send GSS-TSIG unsigned updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IPv4 or IPv6 addresses.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get ad_servers
my $ad_servers = $zone->ad_servers();
# Modify ad_servers
$zone->ad_servers(["1.2.3.4", "dead::beef"]);
```

allow_gss_tsig_for_underscore_zone( )

Use this method to set or retrieve the flag that allows DHCP clients to perform GSS-TSIG signed updates for underscore zones.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get allow_gss_tsig_for_underscore_zone.
my $allow_gss_tsig_for_underscore_zone = $zone->allow_gss_tsig_for_underscore_zone();
# Modify allow_gss_tsig_for_underscore_zone.
$zone->allow_gss_tsig_for_underscore_zone("true");
```

allow_gss_tsig_zone_updates( )

Use this method to enable/disable the zone to accept GSS-TSIG updates. This setting overrides the grid and member GSS-TSIG zone update settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get allow_gss_tsig_zone_updates.
my $allow_gss_tsig_zone_updates = $zone->allow_gss_tsig_zone_updates();
# Modify allow_gss_tsig_zone_updates.
$zone->allow_gss_tsig_zone_updates("true");
```

allow_query( )

Use this method to set or retrieve the allow_query flag of a DNS zone.

You can allow or reject queries from specified IP addresses and networks. You can also specify TSIG keys for the allowed hosts. This setting overrides the member-level query settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IPv4 or IPv6 addresses, networks, and/or Infoblox::DNS::TSIGKey objects. To deny queries from certain IP addresses and networks, add "!" as a prefix to the IP addresses and networks. Specify "any" to allow any query, or undefine to use the Grid::Member::DNS level allow_query list. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting allow_query
my $allow_query = $zone->allow_query();
#Modifying allow_query
$zone->allow_query(["any"]);

#Specifying TSIG keys
my $tsig_key1 = $session->gen_tsig_key();
my $query_key = Infoblox::DNS::TSIGKey->new(
   name => "query_key1",
   key  => $tsig_key1
);
#Modifying allow_query
$zone->allow_query( [ "10.0.0.10" , "20.0.0.0/24","2001::123", $key1,
"2004::444/64","!"."30.0.0.30" , "!"."40.10.0.0/24","!"."2005::222/64"] );
##Un-override allow_query
$zone->allow_query(undef)
```

allow_transfer( )

Use this method to set or retrieve the allow_transfer list of a DNS zone.

You can allow or reject zone transfers from specified IP addresses and networks. This setting overrides the member-level zone transfer settings. You can also provide Transaction Signature (TSIG) keys for clients that are allowed to do zone transfers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IPv4 or IPv6 addresses and networks. It is also possible to specify the TSIG keys of clients that are allowed to perform zone transfers. TSIG keys are used to authenticate zone transfer requests and replies. The same key name and key value must be on the primary and secondary name servers for TSIG-authenticated zone transfers to occur. To deny zone transfers from certain IP addresses and networks, add "!" as a prefix to the IP addresses and networks. Specify "any" to allow any transfer, or undefine to use the Grid::Member::DNS level allow_transfer list. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting allow_transfer
my $allow_transfer = $zone->allow_transfer( );
#Modifying allow_transfer
$zone->allow_transfer(["any"]);
```

```
#Specifying TSIG keys
my $tsig_key1 = $session->gen_tsig_key();
my $key1 = Infoblox::DNS::TSIGKey->new(
   name => "key1",
   key  => $tsig_key1
);
my $key2 = Infoblox::DNS::TSIGKey->new(
   name => "tsig_xfer",
   key  => ":2xCOMPAT"
   );
#Modifying allow_transfer
$zone->allow_transfer(["10.0.0.10", "20.0.0.0/24","2001::1234", $key1, "2001::333/64", $key2,
"!"."30.0.0.30", "!"."40.10.0.0/24","!"."2008::543","!"."2007::456/64"]);
#Use Grid::Member::DNS level allow_transfer list
$zone->allow_transfer(undef)
```

allow_update( )

Use this method to set or retrieve the allow_update flag of a DNS zone.

You can allow or deny dynamic updates from specified IP addresses and networks. This setting overrides member-level update settings. You can also provide TSIG keys for clients that are allowed to do zone updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IPv4 or IPv6 addresses, networks and/or Infoblox::DNS::TSIGKey objects. To deny updates from certain IP addresses and networks, add "!" as a prefix to the IP addresses and networks. Specify "any" to allow any update, or undefine to use the Grid::Member::DNS allow_update list. Note: You should also undefine the update_keys list to use Grid::Member::DNS settings. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting allow_update
my $allow_update = $zone->allow_update();
#Modifying allow_update
$zone->allow_update(["any"]);

#Creating a TSIGKey object.
my $update_key1 = $session->gen_tsig_key();
my $up_key1 = Infoblox::DNS::TSIGKey->new(
   name => "update_key",
   key  => $update_key1
   );

#Modifying allow_update
$zone->allow_update(["10.0.0.10", $up_key1, "20.0.0.0/24","2006::987","2007::777/64",
"!"."30.0.0.30", "!"."40.10.0.0/24"]);

#Use Grid::Member::DNS level allow_update list
$zone->allow_update(undef)
```

allow_update_forwarding( )

Use this method to set or retrieve the allow_update_forwarding flag of a DNS zone.

You can allow or deny dynamic updates forwarded from specified IP addresses and networks. This setting overrides member-level update settings. You can also provide TSIG keys for clients that are allowed to do zone updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IPv4 or IPv6 addresses, networks and/or Infoblox::DNS::TSIGKey objects. To deny updates from certain IP addresses and networks, add "!" as a prefix to the IP addresses and networks. Specify "any" to allow any update, or undefine to use the Grid::Member::DNS allow_update list. Note: You should also undefine the update_keys list to use Grid::Member::DNS settings. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting allow_update_forwarding
my $allow_update_forwarding = $zone->allow_update_forwarding();
#Modifying allow_update_forwarding
$zone->allow_update_forwarding(["any"]);

#Creating a TSIGKey object.
my $update_key1 = $session->gen_tsig_key();
my $up_key1 = Infoblox::DNS::TSIGKey->new(
   name => "update_key",
   key  => $update_key1
   );

#Modifying allow_update_forwarding
$zone->allow_update_forwarding(["10.0.0.10", $up_key1,
"20.0.0.0/24","2009::654/64","!"."30.0.0.30", "!"."40.10.0.0/24","!"."2007::432/64"]);

##Un-override allow_update_forwarding
$zone->allow_update_forwarding(undef);
```

bind_check_names_policy( )

Use this method to set or retrieve the BIND check names policy, which indicates the action the appliance takes when it encounters host names that do not comply with the Strict Hostname Checking policy. This method applies only if the host name restriction policy is set to "Strict Hostname Checking".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The allowed values of the parameter are "fail", "warn" or undefined. This indicates the desired BIND behavior in the case of illegal hostnames. The default value is undefined, which means the appliance ignores illegal hostnames.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bind_check_names_policy value
my $names_policy = $zone->bind_check_names_policy();
#Modify bind_check_names_policy
$zone->bind_check_names_policy("warn");
```

comment( )

Use this method to set or retrieve a descriptive comment about the zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $zone->comment();
#Modify comment
$zone->comment("Modifying the DNS Zone comment");
```

create_underscore_zones( )

Use this method to set or retrieve the create_underscore_zones flag of a DNS zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the create_underscore_zones flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get create_underscore_zones
my $create_underscore_zones = $zone->create_underscore_zones();
#Modify create_underscore_zones
$zone->create_underscore_zones("true");
```

delegate_to( )

Use this method to set or retrieve the delegate_to flag of a DNS zone.

This provides information for the remote name server that maintains data for the delegated zone. The Infoblox appliance redirects queries for data for the delegated zone to this remote name server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Nameserver objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get delegate_to
my $delegate_to = $zone->delegate_to();
#Modify delegate_to
my $nameserver1 = Infoblox::DNS::Nameserver->new(
       name     => "ns1.del.com",
       ipv4addr => "11.11.11.11",
   );
my $nameserver2 = Infoblox::DNS::Nameserver->new(
       name     => "test",
       ipv6addr => "2006::b671",
   );
$zone->delegate_to([$nameserver1, $nameserver2]);
```

delegated_ttl( )

Use this method to set or retrieve the delegated_ttl value of a delegated DNS zone.

You can specify the Time to Live (TTL) values of auto-generated NS and glue records for a delegated zone. This value is the number of seconds that data is cached.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the delegated_ttl in seconds. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. When delegated_ttl is modified, override_delegated_ttl will be set to 'true' implicitly.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get delegated_ttl
my $delegated_ttl = $zone->delegated_ttl();
#Modify delegated_ttl
$zone->delegated_ttl("3600");
```

disable( )

Use this method to set or retrieve the disable flag of a DNS zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $zone->disable();
#Modify disable
$zone->disable("true");
```

disable_forwarding( )

Use this method to set or retrieve the disable_forwarding flag of a DNS zone.

Set this flag to indicate that the name servers hosting the zone should not forward queries that end with the domain name of the zone to any configured forwarders.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to disable forwarding or "false" to enable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable_forwarding
my $disable_forwarding = $zone->disable_forwarding();
#Modify disable_forwarding
$zone->disable_forwarding("true");
```

dnssec_ksk_algorithm()

Use this method to set or retrieve the DNSSEC key-signing key algorithm of the zone.

This property determines which algorithm to use when generating the keys for signing the DNSKEY records.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Currently there are two major versions of the DNSSEC zone encryption mechanism. The difference between the two is in their implementation of the Next Secure (NSEC) records. The older version, which is referred to as NSEC, is based on RFCs 4033-4035. The newer version, which is referred to as NSEC3, is described in RFC 5155. The algorithm field allows you to specify which encryption version to use as well as which encryption algorithm to chose inside the defined NSEC type. Both the ZSK and KSK algorithms should use the same NSEC version; however, they are allowed to use different encryption algorithms within the same NSEC version.

The parameter can be specified as an integer or as a mnemonic. The following table lists the available algorithm values supported by the appliance, their mnemonics, and NSEC type.

```
  Value  Algorithm        Mnemonic
NSEC algorithms:
    1     RSA/MD5         RSAMD5
    3     DSA/SHA-1       DSA
    5     RSA/SHA-1       RSASHA1
NSEC3 algorithms:
    6     DSA/SHA-1       NSEC3DSA
    7     RSA/SHA-1       NSEC3RSASHA1
```

For algorithms not listed above only the mnemonic representation is supported.

The dnssec_ksk_algorithm parameter inherits the grid setting only when it and all the following DNSSEC parameters are undefined: dnssec_ksk_rollover_interval, dnssec_ksk_size, dnssec_signature_expiration, dnssec_zsk_algorithm, dnssec_zsk_rollover_interval, dnssec_zsk_size. When at least one of the parameters is specified, then the dnssec_ksk_algorithm parameter uses the value 'RSASHA1', unless a different value is explicitly specified.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting algorithm
my $algorithm = $zone->dnssec_ksk_algorithm();

# Modifying algorithm
$zone->dnssec_ksk_algorithm("DSA");

#Inheriting the grid DNSSEC settings
$zone->dnssec_ksk_algorithm(undef);
$zone->dnssec_ksk_rollover_interval(undef);
$zone->dnssec_ksk_size(undef);
$zone->dnssec_signature_expiration(undef
$zone->dnssec_zsk_algorithm(undef);
$zone->dnssec_zsk_rollover_interval(undef);
$zone->dnssec_zsk_size(undef);
```

dnssec_ksk_rollover_interval()

Use this method to set or retrieve the rollover interval of the KSK keys.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A positive integer that specifies the number of seconds between KSK rollovers. Normally, this parameter should be larger than the ZSK rollover interval. The DNSSEC key-signing key rollover interval (in seconds) must be between 1 day and seconds to Jan 2, 2038.

The dnssec_ksk_rollover_interval parameter inherits the grid setting only when it and all the following DNSSEC parameters are undefined: dnssec_ksk_algorithm, dnssec_ksk_size, dnssec_signature_expiration, dnssec_zsk_algorithm, dnssec_zsk_rollover_interval, dnssec_zsk_size. When at least one of the parameters is specified, then the dnssec_ksk_rollover_interval parameter uses the value 31536000 seconds (1 year), unless a different value is explicitly specified.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_ksk_rollover_interval
my $ksk_rollover = $zone->dnssec_ksk_rollover_interval();

# Modifying dnssec_ksk_rollover_interval
$zone->dnssec_ksk_rollover_interval(86400);

#Use the grid setting
$zone->dnssec_ksk_algorithm(undef);
$zone->dnssec_ksk_rollover_interval(undef);
$zone->dnssec_ksk_size(undef);
$zone->dnssec_signature_expiration(undef);
$zone->dnssec_zsk_algorithm(undef);
$zone->dnssec_zsk_rollover_interval(undef);
$zone->dnssec_zsk_size(undef);
```

dnssec_ksk_size()

Use this method to set or retrieve the length of the key-signing key.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DNSSEC key-signing key possible size depends on the algorithm. Use the following table to determine the possible values for key sizes:

```
  Algorithm    Key size     Additional restrictions
NSEC
  RSA/MD5      512-4096
  DSA/SHA1     512-1024     Key size must be divisible by 64
  RSA/SHA1     512-4096
  RSA256/SHA2  512-4096
  RSA512/SHA2  1024-4096
NSEC3
  DSA/SHA1     512-1024     Key size must be divisible by 64
  RSA/SHA1     512-4096
  RSA256/SHA2  512-4096
  RSA512/SHA2  1024-4096
```

The dnssec_ksk_size parameter inherits the grid setting only when it and all the following DNSSEC parameters are undefined: dnssec_ksk_algorithm, dnssec_ksk_rollover_interval, dnssec_signature_expiration, dnssec_zsk_algorithm, dnssec_zsk_rollover_interval, dnssec_zsk_size. When at least one of the parameters is specified, then the dnssec_ksk_size parameter uses the value 2048, unless a different value is explicitly specified.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_ksk_size
my $ksk_size = $zone->dnssec_ksk_size();

# Modifying dnssec_ksk_size
$zone->dnssec_ksk_size(4096);

#Use the grid setting
$zone->dnssec_ksk_algorithm(undef);
$zone->dnssec_ksk_rollover_interval(undef);
$zone->dnssec_ksk_size(undef);
$zone->dnssec_signature_expiration(undef);
$zone->dnssec_zsk_algorithm(undef);
$zone->dnssec_zsk_rollover_interval(undef);
$zone->dnssec_zsk_size(undef);
```

dnssec_signature_expiration( )

Use this method to set or retrieve the expiration time for DNSSEC signatures.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The signature expiration time in seconds. The minimum value is 86400 (1 day), the maximum value is the number of seconds from the time you set it to January 2038.

The dnssec_signature_expiration parameter inherits the grid setting only when it and all the following DNSSEC parameters are undefined: dnssec_ksk_algorithm, dnssec_ksk_rollover_interval, dnssec_ksk_size, dnssec_zsk_algorithm, dnssec_zsk_rollover_interval, dnssec_zsk_size. When at least one of the parameters is specified, then the dnssec_signature_expiration parameter uses the value 2592000 seconds (30 days), unless a different value is explicitly specified.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_signature_expiration
my $signature_expiration = $zone->dnssec_signature_expiration();

# Modifying dnssec_signature_expiration
$zone->dnssec_signature_expiration(31536000);

#Use the grid setting
$zone->dnssec_ksk_algorithm(undef);
$zone->dnssec_ksk_rollover_interval(undef);
$zone->dnssec_ksk_size(undef);
$zone->dnssec_signature_expiration(undef);
$zone->dnssec_zsk_algorithm(undef);
$zone->dnssec_zsk_rollover_interval(undef);
$zone->dnssec_zsk_size(undef);
```

dnssec_signed( )

Use this method to check if the zone is DNSSEC signed and to sign/unsign the zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value. Note that for this method it is not necessary to use the call `$session->modify()` on the zone object for the setting to take effect.

Parameter

Set the parameter to "true" to sign the zone, set the parameter to "false" to unsign the zone. The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns "true" if the zone is signed and "false" if it is unsigned.

Example

```
# Checking whether the zone is signed
my $is_signed = $zone->dnssec_signed();

# Signing the zone
$zone->dnssec_signed("true");

# Unsigning the zone
$zone->dnssec_signed("false");
```

dnssec_zsk_algorithm()

Use this method to set or retrieve the DNSSEC zone-signing key algorithm of the zone.

This property determines which algorithm will be used to sign zone data.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Currently there are two major versions of the DNSSEC zone encryption mechanism. The difference between the two is in their implementation of the Next Secure (NSEC) records. The older version, which is referred to as NSEC, is based on RFCs 4033-4035. The newer version, which is referred to as NSEC3, is described in RFC 5155. The algorithm field allows you to specify which encryption version to use as well as which encryption algorithm to chose inside the defined NSEC type. Both the ZSK and KSK algorithms should use the same NSEC version; however, they are allowed to use different encryption algorithms within the same NSEC version.

The parameter can be specified as an integer or as a mnemonic. The following table lists the available algorithm values supported by the appliance, their mnemonics, and NSEC type.

```
   Value  Algorithm        Mnemonic
 NSEC algorithms:
     1     RSA/MD5         RSAMD5
     3     DSA/SHA-1       DSA
     5     RSA/SHA-1       RSASHA1
 NSEC3 algorithms:
     6     DSA/SHA-1       NSEC3DSA
     7     RSA/SHA-1       NSEC3RSASHA1
```

For algorithms not listed above, only the mnemonic representation is supported.

The dnssec_zsk_algorithm parameter inherits the grid setting only when it and all the following DNSSEC parameters are undefined: dnssec_ksk_algorithm, dnssec_ksk_rollover_interval, dnssec_ksk_size, dnssec_signature_expiration, dnssec_zsk_rollover_interval, dnssec_zsk_size. When at least one of the parameters is specified, then the dnssec_zsk_algorithm parameter uses the value 'RSASHA1', unless a different value is explicitly specified.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting algorithm
my $algorithm = $zone->dnssec_zsk_algorithm();

# Modifying algorithm
$zone->dnssec_zsk_algorithm("DSA");

#Inheriting the grid DNSSEC settings
$zone->dnssec_ksk_algorithm(undef);
```

```
$zone->dnssec_ksk_rollover_interval(undef);
$zone->dnssec_ksk_size(undef);
$zone->dnssec_signature_expiration(undef);
$zone->dnssec_zsk_algorithm(undef);
$zone->dnssec_zsk_rollover_interval(undef);
$zone->dnssec_zsk_size(undef);
```

dnssec_zsk_rollover_interval()

Use this method to set or retrieve the zone-signing key rollover interval.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A positive integer that specifies the number of seconds between ZSK rollovers. Normally, this parameter should be smaller than the KSK rollover interval. The DNSSEC zone-signing key over interval (in seconds) must be between 1 day and the number of seconds from present to Jan 2038.

The dnssec_zsk_rollover_interval parameter inherits the grid setting only when it and all the following DNSSEC parameters are undefined: dnssec_ksk_algorithm, dnssec_ksk_rollover_interval, dnssec_ksk_size, dnssec_signature_expiration, dnssec_zsk_algorithm, dnssec_zsk_size. When at least one of the parameters is specified, then the dnssec_zsk_rollover_interval parameter uses the value 2592000 seconds (30 days), unless a different value is explicitly specified.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_zsk_rollover_interval
my $zsk_rollover = $zone->dnssec_zsk_rollover_interval();

# Modifying dnssec_ksk_rollover_interval
$zone->dnssec_zsk_rollover_interval(86400);

#Use the grid setting
$zone->dnssec_ksk_algorithm(undef);
$zone->dnssec_ksk_rollover_interval(undef);
$zone->dnssec_ksk_size(undef);
$zone->dnssec_signature_expiration(undef);
$zone->dnssec_zsk_algorithm(undef);
$zone->dnssec_zsk_rollover_interval(undef);
$zone->dnssec_zsk_size(undef);
```

dnssec_zsk_size()

Use this method to set or retrieve the length of zone-signing key.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DNSSEC zone-signing key size depends on the algorithm. Use the following table to determine the possible values for key sizes:

```
    Algorithm     Key size     Additional restrictions
  NSEC
    RSA/MD5       512-4096
    DSA/SHA1      512-1024     Key size must be divisible by 64
    RSA/SHA1      512-4096
    RSA256/SHA2   512-4096
    RSA512/SHA2   1024-4096
  NSEC3
    DSA/SHA1      512-1024     Key size must be divisible by 64
    RSA/SHA1      512-4096
    RSA256/SHA2   512-4096
    RSA512/SHA2   1024-4096
```

The dnssec_zsk_size parameter inherits the grid setting only when it and all the following DNSSEC parameters are undefined: dnssec_ksk_algorithm, dnssec_ksk_rollover_interval, dnssec_ksk_size, dnssec_signature_expiration, dnssec_zsk_algorithm, dnssec_zsk_rollover_interval. When at least one of the parameters is specified, then the dnssec_zsk_size parameter uses the value 1024, unless a different value is explicitly specified.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_zsk_size
my $zsk_size = $zone->dnssec_zsk_size();

# Modifying dnssec_zsk_size
$zone->dnssec_zsk_size(4096);

#Use the grid setting
$zone->dnssec_ksk_algorithm(undef);
$zone->dnssec_ksk_rollover_interval(undef);
$zone->dnssec_ksk_size(undef);
$zone->dnssec_signature_expiration(undef);
$zone->dnssec_zsk_algorithm(undef);
$zone->dnssec_zsk_rollover_interval(undef);
$zone->dnssec_zsk_size(undef);
```

do_host_abstraction( )

Use this method to set or retrieve the do_host_abstraction property used in zone import operations.

Set this flag to indicate that Infoblox host records will be automatically created from A records when the zone data is imported.

When the `ms_managed()` flag is Stub, Auth_primary, or Auth_both, the `do_host_abstraction()` setting is always false.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to trigger host creation or "false" to disable it. The default value is "false".

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get do_host_abstraction
my $do_host_abstraction = $zone->do_host_abstraction();
#Modify do_host_abstraction
$zone->do_host_abstraction("true");
```

enable_ad_servers( )

Use this method to set or retrieve the flag that allows GSS-TSIG unsigned updates from AD domain controllers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_ad_servers.
my $enable_ad_servers = $zone->enable_ad_servers();
# Modify enable_ad_servers.
$zone->enable_ad_servers("true");
```

email( )

Use this method to set or retrieve the e-mail address of an administrator of a DNS zone.

This setting overrides grid-level e-mail settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The e-mail address of an administrator.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get email
my $email = $zone->email();
#Modify email
$zone->email("admin\@infoblox.com");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $zone->extensible_attributes();
#Modify extensible attributes
$zone->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

forward_only( )

Use this method to set or retrieve the forward_only flag of a DNS forward zone. When you set this flag, the appliance sends queries to forwarders only, and not to other internal or Internet root servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to send queries to forwarders only. Specify "false" to disable it. Default is "false."

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting forward_only
my $forward_only = $fwd_zone->forward_only( );
#Modifying forward_only
$fwd_zone->forward_only("true");
```

forward_to( )

Use this method to set or retrieve the forward_to flag of a DNS zone.

This method provides information for the remote name server to which you want the Infoblox appliance to forward queries for a specified domain name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Member objects for grid secondary servers and Infoblox::DNS::Nameserver objects for external secondary servers. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get forward_to
my $forward_to = $zone->forward_to();
#Modify forward_to
my $member = Infoblox::DNS::Member->new(
       name     => "hostname.com",
       ipv4addr => "3.0.0.2",
   );

my $nameserver = Infoblox::DNS::Nameserver->new(
       name     => "ns1.forwd.com",
       ipv4addr => "11.11.11.11",
   );

$zone->forward_to([$member, $nameserver]);
```

host_name_restriction_policy( )

Use this method to set or retrieve the host_name_restriction_policy flag of a DNS zone.

This setting overrides the member-level host name restriction policy settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

String with the name of predefined record policies or user-defined record policies. The default value is undefined, which is to follow the Record Name policy specified in Infoblox::Grid::Member::DNS level. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get host name restriction policy
my $host_name_restriction_policy = $zone->host_name_restriction_policy();
#Modify host name restriction policy
$zone->host_name_restriction_policy("Allow Any");
```

import_from( )

Use this method to set or retrieve the import_from flag of a DNS zone.

This method specifies the IP address of the Infoblox appliance from which zone data is imported. This is only valid for authoritative zones. Setting this address to '255.255.255.255' and do_host_abstraction to 'true' will create Host records from A records in this zone without importing zone data.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number with the basic format representation of `X.X.X.X` where x refers to decimal digits (8 bits), for example: `128.6.4.7`. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Modify import_from
$zone->import_from("192.168.1.2");
```

initiate_ksk_rollover( )

Use this method to initiate the rollover of the key-signing key for the DNSSEC signed zone.

Parameter

None

Returns

Returns true value upon the successful completion, and returns false value otherwise.

Example

```
#Do KSK rollover
unless($zone->initiate_ksk_rollover()){ warn "KSK rollover failed!";}
```

locked( )

Use this method to set or retrieve the locked flag of a DNS zone.

If you enable this flag other administrators cannot make changes that would "collide". This is for administration purposes only. The zone will continue to serve DNS data even when it is locked.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the locked flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get locked
my $locked = $zone->locked();
#Modify locked
$zone->locked("true");
```

members( )

Use this method to set or retrieve the members flag of a DNS forward zone.

This method specifies the grid member that serves the forward zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Member object(s). The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get members
my $members = $zone->members();
#Modify members
my $memberns1 = Infoblox::DNS::Member->new(
      name     => "member1.forwd.com",
      ipv4addr => "3.3.3.3",
      stealth  => "false",
  );

my $memberns2 = Infoblox::DNS::Member->new(
      name     => "member2.forwd.com",
      ipv4addr => "4.4.4.4",
      stealth  => "false"
  );

$zone->members([$memberns1,$memberns2]);
```

name( )

Use this method to set or retrieve the name flag of a DNS zone. The name must be in Fully Qualified Domain Name (FQDN) format.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Zone name in FQDN (Fully Qualified Domain Name) format. A zone name can have a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $zone->name();
```

network_associations( )

Use this method to retrieve the associated network/network container information. This is a read-only method.

Parameter

No parameter need to be specified

Returns

This method returns an array reference that contains Infoblox::DHCP::Network, Infoblox::DHCP::NetworkContainer, Infoblox::DHCP::IPv6Network, and Infoblox::DHCP::IPv6NetworkContainer objects.

Examples

```
#Getting network associations
my $network_associations = $zone->network_associations();
```

notify_delay( )

Use this method to specify with how many seconds of delay the notify messages are sent to secondaries.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are unsigned integer between 5 and 86400, inclusive. Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting notify_delay
my $notify_delay = $zone->notify_delay( );
#Modifying notify_delay
$zone->notify_delay(5);
```

ms_ad_integrated()

Use this method to set or retrieve the Active Directory-integrated flag of a zone. This attribute is valid only when `ms_managed()` is "Stub", "Auth_primary", or "Auth_both".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the Active Directory-integrated flag or "false" to unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ms_ad_integrated
my $ms_ad_integrated = $zone->ms_ad_integrated();
#Modify ms_ad_integrated
$zone->ms_ad_integrated("true");
```

ms_managed()

Use this method to retrieve the flag that indicates whether the zone is assigned to a Microsoft DNS server. This flag returns the authoritative name server type of the Microsoft DNS server. The value can be:

"None" if the zone is not assigned to any Microsoft DNS server
"Stub" if the zone is assigned to a Microsoft DNS server as a stub zone
"Auth_primary" if only the primary server of the zone is a Microsoft DNS server
"Auth_secondary" if only the secondary server of the zone is a Microsoft DNS server
"Auth_both" if both the primary and secondary servers of the zone are Microsoft DNS servers

Returns

The method returns the attribute value.

Example

```
#Get ms_managed
my $ms_managed = $zone->ms_managed();
```

ms_read_only()

Use this method to retrieve the flag that indicates if a grid member manages the zone served by a Microsoft DNS server in read-only mode. This flag returns "true" when a grid member manages the zone in read-only mode. Otherwise, it returns "false".

When the zone has the `ms_read_only()` flag set to "true", no change can be made to this zone.

Returns

The method returns the attribute value.

Example

```
#Get ms_read_only
my $ms_read_only = $zone->ms_read_only();
```

ms_ddns_mode()

Use this method to set or retrieve the flag that indicates whether an Active Directory-integrated zone with a Microsoft DNS server as primary allows dynamic updates. This flag returns "Secure" if the zone allows secure updates only, "None" if the zone forbids dynamic updates, or "Any" if the zone accepts both secure and nonsecure updates. This flag is valid only if `ms_managed()` is either "Auth_primary" or "Auth_both". If the flag `ms_ad_integrated()` is false, the value "Secure" is not allowed.

Parameter

Specify "Secure", "None", or "Any" to control the dynamic updates that an Active Directory-integrated zone is allowed to accept. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ms_ddns_mode
my $ms_ddns_mode = $zone->ms_ddns_mode();
#Modify ms_ddns_mode
$zone->ms_ddns_mode("None");
```

ms_allow_transfer()

Use this method to set or retrieve the list of DNS clients that are allowed to perform zone transfers from a Microsoft DNS server.

This setting applies only to zones with Microsoft DNS servers that are either primary or secondary servers. This setting does not inherit any value from the grid or from any member that defines an `allow_transfer()` value. This setting does not apply to any grid member. Use the `allow_transfer()` attribute to control allowed clients on grid members.

Valid values are:

An array that contains a list of IPv4 addresses
The keyword "Any" to allow any client

The keyword "NSOnly" to allow only the nameservers listed in the zone
The keyword "None" to deny all zone transfer requests

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify one of the allowed values to identify the DNS clients that are allowed to perform zone transfers from a Microsoft DNS server. The default is "NSOnly".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ms_allow_transfer
my $ms_allow_transfer = $zone->ms_allow_transfer();
#Modify ms_allow_transfer
$zone->ms_allow_transfer("Any");
```

ns_group( )

Use this method to set or retrieve the ns_group flag of a DNS zone.

Specify the name server group that serves DNS for this zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name server group of the zone. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ns_group
my $ns_group = $zone->ns_group();
#Modify ns_group
$zone->ns_group("ns_group_A");
```

override_delegated_ttl( )

Use this method to set or retrieve the override_delegated_ttl flag of a DNS zone.

This flag indicates whether to override the TTL values of auto-generated NS and GLUE records for a delegated zone. If you set it to "true", then you must specify the 'delegated_ttl' value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to override the TTL values or "false" to have these records inherit the TTL values from their parent zone. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_delegated_ttl
my $override_delegated_ttl = $zone->override_delegated_ttl();
#Modify override_delegated_ttl
$zone->override_delegated_ttl("true");
```

override_grid_zone_timer( )

Use this method to set or retrieve the override_grid_zone_timer flag of a DNS zone.

This flag indicates whether to override the grid's zone timer settings. If you set it to "true", then you must specify the following properties: soa_refresh, soa_retry, soa_expire, soa_default_ttl, soa_negative_ttl.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to override grid zone timer settings or "false" to use the grid zone timer settings. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_grid_zone_timer
my $override_grid_zone_timer = $zone->override_grid_zone_timer();
#Modify override_grid_zone_timer
$zone->override_grid_zone_timer("true");
```

override_grid_email( )

Use this method to set or retrieve the override_grid_email flag of a DNS zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to override grid e-mail settings or "false" to use the grid e-mail settings. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_grid_email
my $override_grid_email = $zone->override_grid_email();
#Modify override_grid_email
$zone->override_grid_email("false");
```

override_serial_number( )

Use this method to set the override_serial_number flag of a DNS zone.

If set this flag to 'true', soa_serial_number can be set. This value behaves somewhat unusually in that the server always resets it to undef in the response, even if it is true in the request.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to override serial number. The default value is undef

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_serial_number
my $override_serial_number = $zone->override_serial_number();
#Modify override_serial_number
$zone->override_serial_number("true");
```

override_soa_mname( )

Use this method to set or retrieve the override_soa_mname flag of a DNS zone.

If you set this flag to "true", then you must specify the soa_mname.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the override_soa_mname flag or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_soa_mname
my $override_soa_mname = $zone->override_soa_mname();
#Modify override_soa_mname
$zone->override_soa_mname("true");
```

prefix( )

Use this method to set or retrieve the prefix flag of a DNS zone.

Use this method only when the netmask is greater than 24 bits; that is, for a mask between 25 and 31 bits. Enter a prefix, such as the name of the allocated address block. The prefix can be alphanumeric characters, without blank spaces; for example: 128/26 , 128-189 , or sub-B . This is only valid for authoritative reverse zones.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

This prefix can be anything such as the address range (examples: 0-127, 0/127) or descriptions (examples: first-network, customer1).A prefix can have a maximum of 63 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get prefix
my $prefix = $zone->prefix();
#Modify prefix
$zone->prefix("first-network");
```

primary( )

Use this method to set or retrieve the primary flag of a DNS zone.

This method specifies the grid primary server, external primary servers, or Microsoft DNS servers that are primary servers for the zone. Although a zone typically has just one primary name server, you can specify up to ten independent servers for a single zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an Infoblox::DNS::Member object for a grid primary server, an array reference that contains Infoblox::DNS::Nameserver objects for external primary servers, or Infoblox::DNS::MSServer objects for Microsoft DNS servers. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get primary
my $primary = $zone->primary();
#Modify primary
my $memberns1 = Infoblox::DNS::Member->new(
       name     => "member1.server.com",
       ipv4addr => "1.1.1.1",
   );

$zone->primary($memberns1);
```

remove_sub_zones( )

Use this method to set or retrieve the remove_sub_zones flag of a DNS zone.

You can remove the sub-zones and resource records of the selected zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to remove sub zones or "false" to not to remove sub zones. The default value is undefined, which is to remove all subzones.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get remove_sub_zones
my $remove_sub_zones = $zone->remove_sub_zones();
#Modify remove_sub_zones
$zone->remove_sub_zones("true");
```

secondaries( )

Use this method to set or retrieve the secondaries flag of a DNS zone.

You can specify grid secondary servers, external secondary servers, or Microsoft DNS servers that are secondary servers for the zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Member objects for grid secondary servers, Infoblox::DNS::Nameserver objects for external secondary servers, and Infoblox::DNS::MSServer objects for Microsoft DNS servers. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get secondaries
my $secondaries = $zone->secondaries();
#Modify secondaries
my $memberns1 = Infoblox::DNS::Member->new(
       name     => "member1.server.com",
       ipv4addr => "1.1.1.1",
   );

my $nameserver1 = Infoblox::DNS::Nameserver->new(
       name     => "ns1.exter.com",
       ipv4addr => "11.11.11.11",
   );

$zone->secondaries([$memberns1, $nameserver1]);
```

soa_default_ttl( )

Use this method to set or retrieve the soa_default_ttl flag of a DNS zone.

You can set the Time to Live (TTL) value of the SOA record of a zone. This value is the number of seconds that data is cached.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the soa_default_ttl in seconds. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get soa_default_ttl
my $soa_default_ttl = $zone->soa_default_ttl();
#Modify soa_default_ttl
$zone->soa_default_ttl("3600");
```

soa_expire( )

Use this method to set or retrieve the soa_expire flag of a DNS zone.

This setting defines the amount of time, in seconds, after which the secondary server stops giving out answers about the zone because the zone data is too old to be useful. The default is one week.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the soa_expire in seconds. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get soa_expire
my $soa_expire = $zone->soa_expire();
#Modify soa_expire
$zone->soa_expire("3600");
```

soa_mname( )

Use this method to set or retrieve the soa_mname flag of a DNS zone.

The Infoblox appliance allows you to change the name of the primary server on the SOA record that is automatically created when you initially configure a zone. Use this method to change the name of the primary server on the SOA record. For example, you may want to hide the primary server for a zone. If your device is named dns1.zone.tld, and for security reasons, you want to show a secondary server called dns2.zone.tld as the primary server. To do so, you would go to dns1.zone.tld zone (being the true primary) and change the primary server on the SOA to dns2.zone.tld to hide the true identity of the real primary server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Primary server name in FQDN (Fully Qualified Domain Name) format. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get soa_mname
my $soa_mname = $zone->soa_mname();
#Modify soa_mname
$zone->soa_mname("soamname.com");
```

soa_negative_ttl( )

Use this method to set or retrieve the soa_negative_ttl flag of a DNS zone.

The negative Time to Live (TTL) value of the SOA of the zone indicates how long a secondary server can cache data for "Does Not Respond" responses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the soa_negative_ttl in seconds. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get soa_negative_ttl
my $soa_negative_ttl = $zone->soa_negative_ttl();
#Modify soa_negative_ttl
$zone->soa_negative_ttl("3600");
```

soa_refresh( )

Use this method to set or retrieve the soa_refresh flag of a DNS zone.

This indicates the interval at which a secondary server sends a message to the primary server for a zone to check that its data is current, and retrieve fresh data if it is not.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the soa_refresh in seconds. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get soa_refresh
my $soa_refresh = $zone->soa_refresh();
#Modify soa_refresh
$zone->soa_refresh("1000");
```

soa_retry( )

Use this method to set or retrieve the soa_retry flag of a DNS zone.

This indicates how long a secondary server must wait before attempting to recontact the primary server after a connection failure between the two servers occurs.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the soa_retry in seconds. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get soa_retry
my $soa_retry = $zone->soa_retry();
#Modify soa_retry
$zone->soa_retry("1000");
```

soa_serial_number( )

Use this method to set or retrieve the soa_serial_number flag of a DNS zone.

The serial number in the SOA record incrementally changes every time the record is modified. The Infoblox appliance allows you to change the serial number (in the SOA record) for the primary server so it is higher than the secondary server, thereby ensuring zone transfers come from the primary server (as they should).

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the soa serial number. The default value is 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get soa_serial_number
my $soa_serial_number = $zone->soa_serial_number();
#Modify soa_serial_number
$zone->soa_serial_number("222222");
```

shared_record_groups( )

Use this method to set or retrieve the associated shared record groups of a DNS zone.

If a shared record group is associated with a zone, then all shared records in a shared record group will be shared in the zone. For more information about shared record groups and shared records, please refer to Infoblox Administrator Guide.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference containing the name of shared record groups associated with this zone. The name of shared record group is in string format. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get shared record groups
my $srgs = $zone->shared_record_groups();
#Modify shared record groups. Assume "test_srg" and "another_srg" are the name
#of already defined shared record groups.
$zone->shared_record_groups(["test_srg", "another_srg"]);
```

stub_from( )

Use this method to set or retrieve the stub_from servers of a DNS zone.

This method specifies the primary servers (masters) of the stub zone.

User can also specify grid member server IP/name pair in this list of primary servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Nameserver object(s). The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get stub_from
my $stub_from = $zone->stub_from();
#Modify stub_from
my $nameserver1 = Infoblox::DNS::Nameserver->new(
       name     => "ns1.stub.com",
       ipv4addr => "11.11.11.11",
   );
my $nameserver2 = Infoblox::DNS::Nameserver->new(
       name     => "test",
       ipv4addr => "2006::b671",
   );
#Specify a grid member server IP and name
my $nameserver3 = Infoblox::DNS::Nameserver->new(
       name     => "ns.member.com",
       ipv4addr => "192.168.1.2",
   );
```

```
$zone->stub_from([$nameserver1, $nameserver2, $nameserver3]);
```

stub_members( )

Use this method to set or retrieve the stub_members flag of a DNS zone.

This method specifies the grid member or Microsoft DNS servers of the stub zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Member `object(s)` for grid DNS servers and Infoblox::DNS::MSServer `object(s)` for Microsoft DNS Servers. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get stub_members
my $stub_members = $zone->stub_members();
#Modify stub_members
my $memberns1 = Infoblox::DNS::Member->new(
       name     => "member1.stub.com",
       ipv4addr => "3.3.3.3",
   );

my $memberns2 = Infoblox::DNS::Member->new(
       name     => "member2.stub.com",
       ipv4addr => "4.4.4.4",
   );

$zone->stub_members([$memberns1,$memberns2 ]);
```

views( )

Use this method to set or retrieve the views flag of a DNS zone.

You can list the views in which the zone is located.

When the `ms_managed()` flag is not None, the method is limited to retrieving the list of DNS views only. Setting the DNS view of a zone served by a Microsoft DNS server is done from the Microsoft server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains the Infoblox::DNS::View object. The default value is the "default" view.

Note that the array size must be 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $views = $zone->views();
#Modify views
my $internal_view = Infoblox::DNS::View->new(
               "name"    => "internal",
               "comment" => "insert internal",
               "disable" => "false",
               );

$zone->views([$internal_view]);
```

zone( )

Use this method to retrieve the parent zone value of a DNS zone. A parent zone can be of any type, except that a delegated child zone

can not have a parent zone of delegated type. The zone name can be empty.

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get zone
my $zone1 = $zone->zone();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a zone object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $host_ip =  "192.168.1.2";
my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
    );
unless($session){
        die("Constructor for session failed: ",
               Infoblox::status_code(). ":" . Infoblox::status_detail());
}
print "Session created successfully.\n";
```

#Create a DNS zone.

```
my $memberns1 = Infoblox::DNS::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => $host_ip,
    lead     => "false",
    stealth  => "false"
);

my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name     => "ns1.test.com",
    ipv4addr => "1.1.1.1",
    stealth  => "false",
    TSIGname => "tsig_name",
    TSIGkey  => "ZX1jVJl7C58GT/sc7Q3ucA==",
);

my $nameserver2 = Infoblox::DNS::Nameserver->new(
    name     => "ns2.test.com",
    ipv4addr => "2.2.2.2",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA==",
);

my $default_view = Infoblox::DNS::View->new(name => "default");

#To add zone test.com in the default view
my $firstzone = Infoblox::DNS::Zone->new(
    name        => "test.com",
    views       => [$default_view],
    email       => "admin\@infoblox.com",
    comment     => "add a zone test.com",
    primary     => $memberns1,
    secondaries => [$nameserver1, $nameserver2],
    );
```

```
unless($firstzone){
       die("Construct test.com zone object failed: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
       }
print "test.com zone object created successfully.\n";

$session->add($firstzone)
    or die("Add zone for test.com failed: ",
               $session->status_code(). ":" .$session->status_detail());

print"Zone test.com added successfully.\n";

#Create a reverse zone.
my $revzone1 = Infoblox::DNS::Zone->new(
    name    => "10.1.4.0/24",
    email   => "admin\@infoblox.com",
    comment => "add a zone 4.168.192.in-addr.arpa",
    primary => $memberns1,
    );

unless($revzone1){
     die("Constructor for zone 4.1.10.in-addr.arpa failed: ",
                Infoblox::status_code(). ":" .Infoblox::status_detail());
       }
print "4.1.10.in-addr.arpa zone object created successfully.\n";

$session->add($revzone1)
    or die("Add zone for 4.1.10.in-addr.arpa failed: ",
                $session->status_code(). ":" .$session->status_detail());

print"Zone 4.1.10.in-addr.arpa added successfully.\n";

#Create a reverse zone with a prefix.
my $revzone2 = Infoblox::DNS::Zone->new(
    name    => "10.168.192.in-addr.arpa",
    email   => "admin\@infoblox.com",
    comment => "add a zone 0-63.10.168.192.in-addr.arpa",
    prefix  => "0-63",
    );

unless($revzone2){
        die("Constructor for zone 0-63.10.168.192.in-addr.arpa failed: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
       }
print "0-63.10.168.192.in-addr.arpa zone object created successfully.\n";

$session->add($revzone2)
    or die("Add zone for 0-63.10.168.192.in-addr.arpa failed: ",
               $session->status_code(). ":" .$session->status_detail());

print"Zone 0-63.10.168.192.in-addr.arpa added successfully.\n";
```

#Search for a specific DNS zone

```
#Searches zones that ends with com
my @search_result = $session->search(
    object => "Infoblox::DNS::Zone",
    name   => ".*\\.com"
);

my $search_obj = $search_result[0];
unless($search_obj){
       die("Search zone failed: ",
                $session->status_code() . ":" . $session->status_detail());
       }

print "Search zone object found at least 1 matching entry\n";
```

#Get and modify a DNS zone

```
#Get zone test.com through session
my @result_array = $session->get(
    object => "Infoblox::DNS::Zone",
    name   => "test.com",
    view   => "default"
);
my $for_obj = $result_array[0];
```

```
unless($for_obj){
       die("Get zone test.com failed: ",
       $session->status_code() . ":" . $session->status_detail());
       }
print "Get test.com zone object found at least 1 matching entry\n";

#Modify one of the attributes of the specified zone
$for_obj->comment("this is a modified comment.");

#Apply the change
    $session->modify($result_array[0])
        or die("Modify zone for test.com failed: ",
               $session->status_code(). ":" .$session->status_detail());

 print "test.com zone object modified successfully.\n";

#Get and modify zone 4.1.10.in-addr.arpa.
my @result = $session->get(
    object => "Infoblox::DNS::Zone",
    name   => "4.1.10.in-addr.arpa"
);

my $rev_object = $result[0];
unless ($rev_object) {
       die("Get zone 4.1.10.in-addr.arpa failed: ",
                $session->status_code() . ":" . $session->status_detail());
       }
print "Get 4.1.10.in-addr.arpa zone object found at least 1 matching entry\n";

#Modify one of the attributes of the specified zoe
$rev_object->secondaries([$nameserver2]);

#Apply the changes
$session->modify($rev_object)
            or die("Modify reverse zone 4.1.10.in-addr.arpa failed: ",
                        $session->status_code(). ":" .$session->status_detail());

print "4.1.10.in-addr.arpa zone object modified successfully.\n";
```

#Remove a DNS zone

```
#Remove zone test.com from a view.
$session->remove(
    object => "Infoblox::DNS::Zone",
    name   => "test.com",
    views  => [$default_view]
    )
    or die("Remove zone test.com failed: ",
             $session->status_code(). ":" .$session->status_detail());

print "Remove zone test.com successfull\n";

#Remove zone 4.1.10.in-addr.arpa from a view.
$session->remove(
    object => "Infoblox::DNS::Zone",
    name   => "4.1.10.in-addr.arpa",
    views  => [$default_view]
    )
    or die("Remove zone 4.1.10.in-addr.arpa failed: ",
             $session->status_code(). ":" .$session->status_detail());

print "Remove zone 4.1.10.in-addr.arpa successfull\n";

#Remove zone with prefix from a view.
$session->remove(
    object => "Infoblox::DNS::Zone",
    name    => "10.168.192.in-addr.arpa",
    prefix  => "0-63",
   )
   or die("Remove zone 0-63.10.168.192.in-addr.arpa failed: ",
             $session->status_code(). ":" .$session->status_detail());

print "Remove zone 0-63.10.168.192.in-addr.arpa successfull\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::Member, Infoblox::DNS::Nameserver, Infoblox::DNS::SRG, Infoblox::DNS::TSIGKey, Infoblox::DNS::View, Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid - Manages the Grid settings.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
  - Infoblox::Session->modify( )
- METHODS
  - audit_log_format( )
  - alternate_resolver( )
  - copy_audit_to_syslog( )
  - enable_member_redirect( )
  - enable_ntp( )
  - enable_recycle_bin( )
  - enable_snmpv3_query( )
  - enable_snmpv3_traps( )
  - lcd_input( )
  - name( )
  - nat_group_list( )
  - notification_email_addr( )
  - ntp_server( )
  - ntp_access_list( )
  - ntp_authentication_key( )
  - password_min_length( )
  - prefer_resolver( )
  - query_comm_string ( )
  - remote_console_access( )
  - secret( )
  - support_access( )
  - scheduled_backup( )
  - session_timeout( )
  - snmp_admin( )
  - snmpv3_query_users( )
  - syslog_server( )
  - syslog_size( )
  - trap_comm_string( )
  - trap_receiver( )
  - vpn_port( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid - Manages the Grid settings.

DESCRIPTION

This class manages the database that is distributed among all grid members. The grid object is not intended to be instantiated, it should only be used to modify information.

A grid is a group of two or more Infoblox appliances that share sections of a common, distributed, built-in database and also, are connected together to provide a single point of administration and service configuration in a secure, highly available environment.

CONSTRUCTOR

The Infoblox::Grid object is automatically generated upon the successful installation of the Infoblox appliance. It does not require manual construction.

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a grid object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Example

```
my $grid = $session->get(
    object => "Infoblox::Grid");
```

Infoblox::Session->search( )

Use this method to search for the existing objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Example

```
my $grid = $session->search(
    object => "Infoblox::Grid");
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the name
$grid->name("Infoblox2");
# Submit modification
my $response = $session->modify( $grid );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a grid.

audit_log_format( )

Use this method to set or retrieve the audit log format.

Parameter

Valid values are "detailed" or "brief". Default value is "detailed".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get audit_log_format
my $audit_log_format = $grid->audit_log_format();
#Modify audit_log_format
$grid->audit_log_format("brief");
```

alternate_resolver( )

Use this method to set or retrieve the alternate resolver.

Parameter

Valid data types are an IPv4 address, an Infoblox::DNS::Nameserver object or an Infoblox::DNS::Member object. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get alternate_resolver
my $alternate = $grid->alternate_resolver();
#Modify alternate_resolver
$grid->alternate_resolver("4.4.4.4");
```

copy_audit_to_syslog( )

Use this method to set or retrieve the flag that enables the appliance to copy audit log messages to syslog.

Parameter

Valid values are "true" or "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get copy_audit_to_syslog
my $copy_audit_to_syslog = $obj->copy_audit_to_syslog();
#Modify copy_audit_to_syslog
$obj->copy_audit_to_syslog("true");
```

enable_member_redirect( )

Use this method to set or retrieve the flag that specifies if GUI redirection is enabled for members.

Parameter

Valid values are "true" or "false". Default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_member_redirect
my $enable_member_redirect = $obj->enable_member_redirect();
#Modify enable_member_redirect
$obj->enable_member_redirect("true");
```

enable_ntp( )

Use this method to enable/disable NTP (Network Time Protocol).

Parameter

Specify "true" to enable NTP or "false" to disable it. Default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get ntp
my $ntp = $grid->enable_ntp();
#Modify NTP
$grid->enable_ntp ("true");
```

enable_recycle_bin( )

Use this method to enable/disable the recycle bin.

When the recycle bin is enabled, deleted DNS data (Zone,View) and DHCP data (Network,Network container,Shared network,DHCP range,Fixed address,DHCP MAC filter) is moved to the recycle bin and can be restored later.

Parameter

Specify "true" to enable the recycle bin or "false" to disable it. Default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get enable_recycle_bin
my $recycle_bin = $grid->enable_recycle_bin();
#Modify enable_recycle_bin
```

```
$grid->enable_recycle_bin ("false");
```

enable_snmpv3_query( )

Use this method to enable or disable SNMPv3 queries.

Parameter

Specify "true" to enable SNMPv3 queries or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get enable_snmpv3_query
my $enable_snmpv3_query = $grid->enable_snmpv3_query();
#Modify enable_snmpv3_query
$grid->enable_snmpv3_query("false");
```

enable_snmpv3_traps( )

Use this method to enable or disable SNMPv3 traps.

Parameter

Specify "true" to enable SNMPv3 traps or "false" to disable it. Default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get enable_snmpv3_traps
my $enable_snmpv3_traps = $grid->enable_snmpv3_traps();
#Modify enable_snmpv3_traps
$grid->enable_snmpv3_traps("true");
```

lcd_input( )

Use this method to enable/disable LCD (Liquid Crystal Display) input. Default value is "true".

Parameter

Specify "true" to enable LCD input or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get lcd_input
my $lcd = $grid->lcd_input();
#Modify lcd_input
$grid->lcd_input("false");
```

name( )

Use this method to set or retrieve the grid name.

Parameter

String with the name of the grid.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get name
my $name = $grid->name();
#Modify name
$grid->name("InfobloxGrid");
```

nat_group_list( )

Use this method to set or retrieve the NAT (Network Address Translation) group list.

Parameter

Array reference that contains each Infoblox::Grid::NatGroup object. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Add a new nat_group_list
my $nat1 = Infoblox::Grid::NatGroup->new( "name" => "natgroup" ,
                                          "comment" => "new NAT group" );

$grid->nat_group_list( [$nat1] );

#To undefined the nat_group_list
$grid->nat_group_list([ ]);
```

notification_email_addr( )

Use this method to set or retrieve the notification e-mail address.

Parameter

Text with the e-mail address of the administrator.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get notification_email_addr
my $email = $grid->notification_email_addr();
#Modify notification_email_addr
$grid->notification_email_addr("admin\@domain.com");
```

ntp_server( )

Use this method to set or retrieve the NTP (Network Time Protocol) server list. Only valid when enable_ntp is set to "true".

Parameter

Array reference that contains Infoblox::Grid::NTPServer objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Add a new ntp_server
my $ntp_server1 = Infoblox::Grid::NTPServer->new(address => "1.1.1.1");
my $ntp_server2 = Infoblox::Grid::NTPServer->new(address => "2.2.2.2");
$grid->ntp_server([$ntp_server1, $ntp_server2]);
```

ntp_access_list( )

Use this method to set or retrieve the NTP (Network Time Protocol) access control list. Only valid when enable_ntp is set to "true".

Parameter

Valid value is an array reference that contains Infoblox::Grid::NTPAccess objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get ntp_access_list
my @ntp_access_list = $grid->ntp_access_list();
#Modify ntp_access_list
```

```
$grid->ntp_access_list([$ntpac1,$ntpac2]);
#To undefined ntp_access_list
$grid->ntp_access_list([ ]);
```

ntp_authentication_key( )

Use this method to set or retrieve the NTP (Network Time Protocol) authentication keys of the grid. Only valid when enable_ntp is set to "true".

Parameter

Valid value is an array reference that contains Infoblox::Grid::NTPKey objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Add a new ntp_authentication_key
my $ntp_key1 = Infoblox::Grid::NTPKey->new(
           key_number => "1111",
           key_type   => "M",
           key_string => "aaaa"
       );

my $ntp_key2 = Infoblox::Grid::NTPKey->new(
           key_number => "2222",
           key_type   => "n",
           key_string => "d3e54352e5548080"
       );

$grid->ntp_authentication_key([$ntp_key1, $ntp_key2]);
#To undefined ntp_authentication_key
$grid->ntp_authentication_key(undef);
```

password_min_length( )

Use this method to set or retrieve the minimum user password length.

Parameter

Valid values are between 4 and 64.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get password_min_length
my $password_min_length = $obj->password_min_length();
#Modify password_min_length
$obj->password_min_length(6);
```

prefer_resolver( )

Use this method to set or retrieve the preferred resolver.

Parameter

Valid data types are an IPv4 address, an Infoblox::DNS::Nameserver object or an Infoblox::DNS::Member object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get prefer_resolver
my $prefer_resolver = $grid->prefer_resolver();
#Modify prefer_resolver
$grid->prefer_resolver("3.3.3.3");
```

query_comm_string ( )

Use this method to set or retrieve the SNMP (Simple Network Management Protocol) community string that management systems must send together with their queries to the Infoblox appliance. If this is set to undef, SNMP queries will be disabled.

Parameter

Text with the SNMP community string.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get query_comm_string
my $query_comm_string = $grid->query_comm_string();
#Modify query_comm_string
$grid->query_comm_string("private");
```

remote_console_access( )

Use this method to enable/disable remote console access.

Parameter

Specify "true" to enable remote console access or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get remote_console_access
my $remote_console_access = $grid->remote_console_access();
#Modify remote_console_access
$grid->remote_console_access("false");
```

secret( )

Use this method to set or retrieve the shared secret of the grid.

Parameter

Text with the shared secret of the grid.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get secret
my $secret = $grid->secret();
#Modify secret
$grid->secret("shared secret");
```

support_access( )

Use this method to enable/disable support access. Default value is "true".

Parameter

Specify "true" to enable support access or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get support_access
my $support_access = $grid->support_access();
#Modify support_access
$grid->support_access("false");
```

scheduled_backup( )

Use this method to retrieve the scheduled backup configuration.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an Infoblox::Grid::ScheduledBackup object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get scheduled_backup
my $scheduled_backup = $grid->scheduled_backup();
#Modify scheduled_backup
$grid->scheduled_backup($scheduled_backup);
```

session_timeout( )

Use this method to set or retrieve the session timeout.

Parameter

Number that specifies the session timeout for the grid (range is from 60 to 31536000 seconds).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get session_timeout
my $session_timeout = $grid->session_timeout();
#Modify session_timeout
$grid->session_timeout("6000");
```

snmp_admin( )

Use this method to set or retrieve the SNMP admin object.

Parameter

Valid data type is an Infoblox::Grid::SNMP::Admin object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Add a SNMP Admin object
my $snmp = Infoblox::Grid::SNMP::Admin->new(
    sysContact  => "Somebody",
    sysLocation => "Infoblox Headquarter",
    sysName     => "noc1",
    sysDescr    => "SunOS noc1 5.6 Generic_105181-03 sun4u"
);

$grid->snmp_admin($snmp);
#To undefined SNMP Admin object
$grid->snmp_admin(undef);
```

snmpv3_query_users( )

Use this method to set or retrieve the SNMPv3 users from which the appliance accepts SNMP queries.

Parameter

Valid value is a reference to the list of the Infoblox::Grid::SNMP::User manpage objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get snmpv3_query_users field
my $snmpv3_users = $grid->snmpv3_query_users();
my $user = $$snmpv3_users[0];
```

```
#Set snmpv3_query_users field
$grid->snmpv3_query_users([$user1, $user2]);
```

syslog_server( )

Use this method to set or retrieve the syslog server.

Parameter

Array reference that contains Infoblox::Grid::SyslogServer objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
my $syslog1 = Infoblox::Grid::SyslogServer->new(
    ipv4addr => "44.44.44.44",
    severity => "debug",
);

my $syslog2 = Infoblox::Grid::SyslogServer->new(
    ipv4addr => "55.55.55.55",
    severity => "debug",
);

$grid->syslog_server([$syslog1, $syslog2]);
```

syslog_size( )

Use this method to set or retrieve the grid syslog rotation size in megabytes.

Parameter

Valid value is between 10 and 300.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get syslog_size
my $syslog_size = $obj->syslog_size();
#Modify syslog_size
$obj->syslog_size(100);
```

trap_comm_string( )

Use this method to set or retrieve the trap community string. If this is set to undef, SNMP traps will be disabled.

Parameter

Text with the SNMP trap community string.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get trap_comm_string
my $trap_comm_string = $grid->trap_comm_string();
#Modify trap_comm_string
$grid->trap_comm_string("public");
```

trap_receiver( )

Use this method to set or retrieve a trap receiver.

Parameter

Array reference that contains IPv4 addresses (example: 192.168.1.2) and the Infoblox::Grid::SNMP::TrapReceiver manpage objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get trap_receiver
my $trap_receiver = $grid->trap_receiver();
#Modify trap receiver (SNMPv2)
$grid->trap_receiver(["10.0.0.10"]);
#SNMPv3 trap receivers
my $receiver1 = Infoblox::Grid::SNMP::TrapReceiver->new( ip_addr => "19.20.21.22", user => $user);
$grid->trap_receiver([$receiver1]);
```

vpn_port( )

Use this method to set or retrieve the VPN port.

Parameter

Number of the Virtual Private Network (VPN) port of the grid. Value must be between 1024 and 63999.

Returns

If you specified a parameter,the method returns true when the modification succeeds,and returns false when the operation fails.

Example

```
#Get port
my $port = $grid->vpn_port();
#Modify vpn_port
$grid->vpn_port("2000");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object, such create, modify, and remove an object. This sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Get and modify the Grid object.

```
#Get a Grid through the session
my $grid = $session->get(
    object => "Infoblox::Grid",
    name   => "Infoblox"
);
unless ($grid) {
    die("Get Grid failed: ",
        $session->status_code() . ":" . $session->status_detail());
}

#Modifying the value of the specified object.
my $nat1 = Infoblox::Grid::NatGroup->new( "name" => "natgroup" ,
                                          "comment" => "new NAT group" );

$grid->nat_group_list( [$nat1] );
$grid->enable_ntp ("true");
my $ntp_server1 = Infoblox::Grid::NTPServer->new(address => "2.2.2.2");
$grid->ntp_server([$ntp_server1]);

my $ntpac1 = Infoblox::Grid::NTPAccess->new(
                                            address => "2.2.2.2",
```

```
                                          service => "Time",
                                         );

my $ntpac2 = Infoblox::Grid::NTPAccess->new(
                                          address => "4.4.4.4",
                                          service => "Time and NTP control",
                                         );

$grid->ntp_access_list([$ntpac1,$ntpac2]);
my $ntp_key1 = Infoblox::Grid::NTPKey->new(
          key_number => "1111",
          key_type   => "M",
          key_string => "aaaa"
       );
my $ntp_key2 = Infoblox::Grid::NTPKey->new(
          key_number => "2222",
          key_type   => "n",
          key_string => "d3e54352e5548080"
       );
$grid->ntp_authentication_key([$ntp_key1, $ntp_key2]);

#Apply the changes
$session->modify($grid)
    or die("Modify Grid failed: ",
          $session->status_code() . ":" . $session->status_detail());
print "Grid object modified successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::DNS::Zone, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->search(), Infoblox::Session->add(), Infoblox::Session->remove(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::AdAuthServer - Active Directory Authentication Server object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Grid::Admin::AdAuthService->ad_auth_server_list()
- METHODS
  - comment( )
  - disabled( )
  - encryption( )
  - name( )
  - port( )
  - mgmt_port( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::AdAuthServer - Active Directory Authentication Server object

DESCRIPTION

Represents an AD authentication server (i.e., domain controller) that is used to authenticate administrators.

CONSTRUCTOR

```
my $ad_auth_server = Infoblox::Grid::Admin::AdAuthServer->new(
    name        => $fqdn | $ipv4addr,     # Required
    port        => $num,                  # Required
    comment     => $string,               # Optional / Default is empty
    disabled    => "true" | "false",      # Optional / Default is "false"
    encryption  => "SSL" | "None",        # Optional / Default is "None"
    mgmt_port   => "true" | "false",      # Optional / Default is "false"
);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to an object.

Infoblox::Grid::Admin::AdAuthService->ad_auth_server_list()

Use this method to specify AD authentication servers that are used to authenticate administrators.

Example

```
# Construct an object
my $ad_auth_server1 = Infoblox::Grid::Admin::AdAuthServer->new(
    name       => "1.2.3.4",
    port       => 636,
    comment    => "domain controller in room 1"
    disabled   => "false",
    encryption => "SSL",
    mgmt_port  => "false",
);

# Construct another object
my $ad_auth_server2 = Infoblox::Grid::Admin::AdAuthServer->new(
    name       => "foo.com",
    port       => 389,
    comment    => "domain controller in room 2"
    disabled   => "false",
    encryption => "None",
    mgmt_port  => "false",
);

# Configure AdAuthServer
my $response = $ad_auth_service->ad_auth_server_list([$ad_auth_server1, $ad_auth_server2]);
```

METHODS

Use the following methods to access the attributes of an Infoblox::Grid::Admin::AdAuthServer object. Specify a parameter to set the attribute's value, or omit the parameter to get the attribute's value.

comment( )

Use this method to set or retrieve comments about the server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Comments in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $val = $obj->comment();
#Modify comment
$obj->comment("head office");
```

disabled( )

Use this method to disable the Active Directory authentication service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to disable the service. Set the parameter to "false" to enable the service. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disabled
my $val = $obj->disabled();
#Modify disabled
$obj->disabled("true");
```

encryption( )

Use this method to set or retrieve the encryption flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "SSL" to use SSL encryption or "None" to use no encryption.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get encryption
my $val = $obj->encryption();
#Modify encryption
$obj->encryption("SSL");
```

name( )

Use this method to set or retrieve the FQDN or IPv4 address of the AD authentication server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The FQDN or IPv4 address of the AD authentication server in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $val = $obj->name();
#Modify name
$obj->name("1.2.3.5");
```

port( )

Use this method to set or retrieve the port number on the AD server to which the appliance sends authentication requests.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

TCP port of the AD authentication server

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get port
my $val = $obj->port();
#Modify port
$obj->port(389);
```

mgmt_port( )

Use this method to set or retrieve the MGMT port flag of the AD server. Set this flag to "true" to restrict the appliance to connecting to the AD server through the management port only.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the mgmt_port flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mgmt_port
my $val = $obj->mgmt_port();
#Modify mgmt_port
$obj->mgmt_port('false');
```

AUTHOR

Infoblox Inc. http://www.infoblox.com

SEE ALSO

Infoblox::Grid::Admin::AdAuthService, Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::AdAuthService - Active Directory Authentication Service object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
- METHODS
  - ad_auth_server_list( )
  - ad_domain( )
  - comment( )
  - disabled( )
  - name( )
  - timeout( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::AdAuthService - Active Directory Authentication Service object

DESCRIPTION

This object allows you to specify an Active Directory (AD) authentication method and the AD authentication servers that Infoblox uses to authenticate administrators.

CONSTRUCTOR

```
my $obj = Infoblox::Grid::Admin::AdAuthService->new(
   ad_auth_server_list => [$ad_server1, $ad_server2, ...],     #Required
   ad_domain           => $string,                             #Required
   name                => $string,                             #Required
   comment             => $string,                             #Optional / Default is empty
   disabled            => "true" | "false",                    #Optional / Default is "false"
   timeout             => $num,                                #Optional / Default is 5
);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to an object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
# Construct an object
my $ad_auth_server1 = Infoblox::Grid::Admin::AdAuthServer->new(
    name       => "1.2.3.4",
    port       => 636,
    comment    => "domain controller in room 1"
    disabled   => "false",
    encryption => "SSL",
    mgmt_port  => "false",
);

# Construct another object
my $ad_auth_server2 = Infoblox::Grid::Admin::AdAuthServer->new(
    name       => "foo.com",
    port       => 389,
    comment    => "domain controller in room 2"
    disabled   => "false",
    encryption => "None",
```

```
    mgmt_port  => "false",
);

# Construct an object
my $obj = Infoblox::Grid::Admin::AdAuthService->new(
   ad_auth_server_list => [$ad_auth_server1, $ad_auth_server2],
   ad_domain           => "authdhcptest.infoblox.com",
   comment             => "test comment",
   disabled            => "false",
   name                => "Test AD Auth",
   timeout             => 333,
);
# Submit for addition
my $response = $session->add( $obj );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

Apply the following attributes to get a specific object:

name - Required. AD authentication service name

Example

```
my @retrieved_objs = $session->get(
  object => "Infoblox::Grid::Admin::AdAuthService",
    name => "Test AD Auth",
);
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Modify the comment
$obj->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $obj);
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Get the objects with the same name
my @retrieved_objs = $session->get(
  object => "Infoblox::Grid::Admin::AdAuthService",
    name => "Test AD Auth",
);
#Find the desired object from the retrieved list.
my $desired_obj = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_obj );
```

METHODS

Use the following methods to access the attributes of an Infoblox::Grid::Admin::AdAuthService object. Specify a parameter to set the attribute's value, or omit the parameter to get the attribute's value.

ad_auth_server_list( )

Use this method to set or retrieve the AD authentication server list.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains a list of Infoblox::Grid::Admin::AdAuthServer objects that describe the AD servers

to which the appliance connects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ad_auth_server_list
my $val = $obj->ad_auth_server_list();
#Modify ad_auth_server_list
$obj->ad_auth_server_list([ $adauthserver1 ]);
```

ad_domain( )

Use this method to set or retrieve the Active Directory domain in which the user credentials are located.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid Active Directory domain name

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ad_domain
my $val = $obj->ad_domain();
#Modify ad_domain
$obj->ad_domain("authdhcptest.infoblox.com");
```

comment( )

Use this method to set or retrieve comments about the service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Comments in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $val = $obj->comment();
#Modify comment
$obj->comment("head office");
```

disabled( )

Use this method to disable the Active Directory authentication service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to disable the service. Set the parameter to "false" to enable the service. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disabled
my $val = $obj->disabled();
#Modify disabled
$obj->disabled("true");
```

name( )

Use this method to set or retrieve the name of the AD authentication service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The AD authentication service name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $val = $obj->name();
#Modify name
$obj->name("AD service");
```

timeout( )

Use this method to set or retrieve the timeout value, which is the number of seconds that the appliance waits for a response from the AD server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of seconds that the Infoblox appliance waits to connect to the server. The default value is 5.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get timeout
my $val = $obj->timeout();
#Modify timeout
$obj->timeout(22);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an Active Directory Authentication Service object, such as add, get, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to an AD Auth Service object insertion

```
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create an AD Auth Service object

```
# Construct an object
```

```
my $ad_auth_server1 = Infoblox::Grid::Admin::AdAuthServer->new(
    name       => "1.2.3.4",
    port       => 636,
    comment    => "domain controller in room 1"
    disable    => "false",
    encryption => "SSL",
    mgmt_port  => "false",
);

# Construct another object
my $ad_auth_server2 = Infoblox::Grid::Admin::AdAuthServer->new(
    name       => "foo.com",
    port       => 389,
    comment    => "domain controller in room 2"
    disable    => "false",
    encryption => "None",
    mgmt_port  => "false",
);

#Create an AD Auth Service object
my $ad_service = Infoblox::Grid::Admin::AdAuthService->new(
    ad_auth_server_list => [ $ad_auth_server1, $ad_auth_server2 ],
    ad_domain           => "authdhcptest.infoblox.com",
    comment             => "test comment",
    disabled            => "false",
    name                => "Test AD Auth",
    timeout             => 333,
);
unless($ad_service) {
       die("Construct AD Auth Service object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "AD Auth Service object created successfully\n";

#Add the AD Auth Service object into the Infoblox appliance through a session
$session->add($ad_service)
    or die("Add AD Auth Service object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "AD Auth Service object added successfully\n";
```

#Get and modify an AD Auth Service

```
#Get an AD Auth Service through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::AdAuthService",
    name => "Test AD Auth",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get AD Auth Service failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get AD Auth Service object found at least 1 matching entry\n";

#Modify one of the attributes of the specified AD Auth Service
$object->comment ("modified comment");

#Apply the changes
$session->modify($object)
    or die("Modify AD Auth Service failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "AD Auth Service object modified successfully \n";
```

#Remove an AD Auth Service

```
#Get an AD Auth Service through the session
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::AdAuthService",
    name       => "Test AD Auth"
);

my $object = $retrieved_objs[0];
unless ($object) {
       die("Get AD Auth Service failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get AD Auth Service object found at least 1 matching entry\n";
```

```
#submit the object for removal
$session->remove($object)
    or die("Remove AD Auth Service failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "AD Auth Service object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com

SEE ALSO

Infoblox::Grid::Admin::AdAuthServer, Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::AuthPolicy - Authentication Policy object

- NAME
- DESCRIPTION
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - admin_groups( )
  - auth_services( )
  - default_group( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::AuthPolicy - Authentication Policy object

DESCRIPTION

This object represents the grid authentication policy.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to the Grid Authentication Policy object.

Infoblox::Session->get( )

Use this method to retrieve the grid authentication policy from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

No parameters are required.

Example

```
my $auth_policy = $session->get(
        object => "Infoblox::Grid::Admin::AuthPolicy",
    );
```

Infoblox::Session->modify( )

Use this method to modify the grid authentication policy on the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Changing the default group
$auth_policy->default_group($group);
#Submit modification
my $response = $session->modify( $auth_policy );
```

METHODS

This section describes all the methods that you can use to set or retrieve the attribute values of the grid authentication policy.

admin_groups( )

Use this method to set or retrieve the list of local admin groups that are mapped to remote administration groups. The default value is an empty list.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::Grid::Admin::Group/> objects.

Returns

If you specified a parameter, the method returns "true" when the modification succeeds, and returns "false" when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the admin groups list
my $admin_groups = $auth_policy->admin_groups();

#Modify the admin_groups
$auth_policy->admin_groups([$admingroup1,$admingroup2]);
```

auth_services( )

Use this method to set or retrieve the list of authentication services that grid members use to authenticate admins. The default is a list that contains the LocalUserAuthService object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains an ordered list of Infoblox::Grid::Admin::AdAuthService/>, Infoblox::Grid::Admin::LocalUserAuthService/> and Infoblox::Grid::Admin::RadiusAuthService/> objects.

Returns

If you specified a parameter, the method returns "true" when the modification succeeds, and returns "false" when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the services list
my $auth_services = $auth_policy->auth_services();
#Modify the auth_services
$auth_policy->auth_services([$radius1, $local, $ad2]);
```

default_group( )

Use this method to set or retrieve the default admin group that provides authentication in case no valid group is found. The default value is none (represented by undef).

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an Infoblox::Grid::Admin::Group/> object.

Returns

If you specified a parameter, the method returns "true" when the modification succeeds, and returns "false" when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the default group
my $default_group = $auth_policy->default_group();

#Modify the default_group
$auth_policy->default_group($group2);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as get and modify. This sample code also includes error handling for the operations.

#Preparation prior to getting and modifying an auth policy object

```
use strict;
use Infoblox;
```

```
#Creating a session to appliance.
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
    );
unless( $session ){
    die("Constructor for session failed:",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print" Session object created successfully \n";
```

#Retrieving the auth policy object my $policy = $session->get(object => 'Infoblox::Grid::Admin::AuthPolicy',) || die("Retrieval of the auth policy failed: ", Infoblox::status_code() . ":" . Infoblox::status_detail());

```
B<#Retrieving the local user auth service>
 my $local_auth = $session->get(
                              object => 'Infoblox::Grid::Admin::LocalUserAuthService',
                             ) || die("Retrieval of the local user auth service failed: ",
                 Infoblox::status_code() . ":" . Infoblox::status_detail());


B<#Creation of a Radius auth server/service pair>
 my $ads = Infoblox::Grid::Admin::RadiusAuthServer->new(
                                                        fqdn_or_ip => '10.1.0.1',
                                                        shared_secret => 'secret',
                                                       ) ||
   die("Couldn't create the auth server: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());


 my $service = Infoblox::Grid::Admin::RadiusAuthService->new(
                                                             acct_timeout => 1100,
                                                             auth_timeout => 1100,
                                                             name => 'some.name.com',
                                                             radius_servers => [$ads]
                                                            ) ||
   die("Couldn't create the auth service: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());


 $session->add($service) || die("Couldn't add the auth service: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());


B<#Retrieving the auth service created above>
 my $radius_auth = $session->get(
                              object => 'Infoblox::Grid::Admin::RadiusAuthService',
                              name   => 'some.name.com',
                         ) || die("Retrieval of the radius auth service failed: ",
                 Infoblox::status_code() . ":" . Infoblox::status_detail());


B<#Modify the authpolicy>
 $policy->auth_services([$radius_auth,$local_auth]);


 $session->modify($policy) || die("Policy modify failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Admin::Group/>, Infoblox::Grid::Admin::AdAuthService/>, Infoblox::Grid::Admin::LocalUserAuthService/>, Infoblox::Grid::Admin::RadiusAuthService/>, Infoblox::Session->get(), Infoblox::Session->get()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::Group - Admin Group object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - access_method( )
  - comment( )
  - disabled( )
  - extensible_attributes( )
  - name( )
  - superuser( )
  - roles( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::Group - Admin Group object.

DESCRIPTION

An Admin Group object creates and manages a local admin group on the Infoblox appliance. The privileges and properties that are set for the group apply to all the admin accounts that are assigned to the group.

CONSTRUCTOR

```
 my $group = Infoblox::Grid::Admin::Group->new(
     name                  => $string,                                                    #Required
     access_method         => ["GUI"|"API",...],
#Optional / Default is ["GUI", "API"]
     comment               => $string,                                                    #Optional
/ Default is undefined
     disabled              => "true" | "false",
#Optional / Default is "false"
     extensible_attributes => { $string => $string | $num, $string => [ $string | $num, ... ], ... }
#Optional / Default is undefined
     superuser             => "true" | "false",
#Optional / Default is "false"
     roles                 => [$string, ...]
#Optional / Default is an empty array reference
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to an Admin Group object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the Group object
my $group = Infoblox::Grid::Admin::Group->new(
    name                  => "testgroup",
    access_method         => ["GUI", "API"],
    comment               => "test group",
    disabled              => "true",
    superuser             => "false"
);
# Submit for addition
```

```
my $response = $session->add( $group );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific Group object:

 name      - Required. Group name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::Group",
    name       => "testgroup" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment.
$group->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $group );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the group objects with the same starting name
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::Group",
    name       => "testgroup" );
# find the desired object from the retrieved list.
my $desired_group = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_group );
```

Infoblox::Session->search( )

Use this method to search for Admin Group objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific Group object:

 name      - Optional. Group name in string format (regular expression).
 roles     - Optional. An array of role names in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all Group objects that start with "test"
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::Admin::Group",
    name   => "test.*" );

# search for all Group objects that start with "test"
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::Admin::Group",
    roles   => ["DHCP Admin", "AAA Admin"]);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a Group object.

access_method( )

Use this method to set or retrieve a group's access methods. Access methods specify whether a group can use the GUI and the API to access the appliance. Valid value is an array reference that contains the literal string "GUI" and/or "API". The default value is ["GUI", "API"].

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference of the literal string "GUI" and/or "API". It indicates whether a group has permission to access the appliance through the GUI and/or API.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get access methods
my $ref_access_method = $group->access_method();
#Modify access methods
$group->access_method(["GUI"]);
```

comment( )

Use this method to set or retrieve the comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Pertinent information about the admin group, such as location or department. Comment in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $group->comment();
#Modify comment
$group->comment("This is the modified comment for admin group");
```

disabled( )

Use this method to set or retrieve the "disabled" flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The admin group is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disabled
my $disabled = $group->disabled();
#Modify disabled
$group->disabled("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a Group object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $group->extensible_attributes();
#Modify extensible attributes
$group->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the name of an admin group.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name of the admin group.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $group->name();
```

superuser( )

Use this method to set or retrieve the "superuser" flag. A superuser group can perform all operations on the appliance, and can view and configure all types of data.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The default value for this field is "false" which means that this is not a superuser admin group. The admins in the group have limited privileges.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get super user flag
my $disabled = $group->superuser();
#Modify super user
$group->superuser("true");
```

roles( )

Use this method to set or retrieve the role names associated with this admin group.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The default value for this field is an empty array. A valid value is an array reference with literal strings containing valid role

names. The corresponding role objects must exist in the database.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get roles
my $ref_roles = $group->roles();
#Modify roles
$group->roles(["role1", "role2"]);
#Remove roles
$group->roles([]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to an Admin Group object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $role = Infoblox::Grid::Admin::Role->new(
    name                => "testrole",
    comment             => "test admin role",
);

unless ($role) {
       die("Construct role object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Admin Role object created successfully\n";

#Add the Admin Role object to the Infoblox appliance through a session
$session->add($role)
       or die("Add Admin Role object failed: ",
                     $session->status_code() . ":" . $session->status_detail());
print "Admin Role object added to server successfully\n";
```

#Create an Admin Group object

```
my $group = Infoblox::Grid::Admin::Group->new(
    name                => "testgroup",
    comment             => "test admin group",
    disabled            => "false",
    superuser           => "false",
    access_method       => ["GUI", "API"],
    roles               => ["testrole"]
);

unless ($group) {
       die("Construct group object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Admin Group object created successfully\n";

#Add the Admin Group object to the Infoblox appliance through a session
```

```
$session->add($group)
       or die("Add Admin Group object failed: ",
                        $session->status_code() . ":" . $session->status_detail());
print "Admin Group object added to server successfully\n";
```

#Search for an Admin Group

```
my @retrieved_objs = $session->search(
               object => "Infoblox::Grid::Admin::Group",
               name   => "test.*"
               );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search Admin Group object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search Admin Group object found at least 1 matching entry\n";
```

#Get and modify an Admin Group object

```
#Get the Admin Group object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::Group",
    name   => "testgroup"
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get Admin Group object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin Group object found at least 1 matching entry\n";

#Modify the Admin Group object
$object->name("testgroup_renamed");
$object->access_method(["API"]);
$object->comment("modified test group");
$object->disabled("false");
$object->superuser("false"); # no real change

#Apply the changes.
$session->modify($object)
       or die("Modify Admin Group object failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "Admin Group object modified successfully \n";
```

#Remove an Admin Group object

```
#Get the Admin Group object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::Group",
    name   => "testgroup"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Admin Group object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin Group object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove Admin Group object failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "Admin Group object removed successfully \n";
```

#Clean up the Admin Role object

```
$session->remove($role)
       or die("Remove Admin Role object failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "Admin Role object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Admin::Role, Infoblox::Grid::Admin::User, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::LocalUserAuthService - Local User Authentication Service object

- NAME
- DESCRIPTION
- MODULE METHODS
  - Infoblox::Grid::Admin::AuthPolicy->auth_services( )
- SESSION METHODS
  - Infoblox::Session->get( )
- METHODS
  - comment( )
  - disabled( )
  - name( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::LocalUserAuthService - Local User Authentication Service object

DESCRIPTION

This object represents a local authentication service for authenticating users against the local database. All methods are read-only.

MODULE METHODS

The following functions can be applied to an Infoblox::Grid::Admin::LocalUserAuthService object.

Infoblox::Grid::Admin::AuthPolicy->auth_services( )

Use this function to set or retrieve a reference to an array of authentication services.

Example

```
#Retrieve the local user authentication service
my $local = $session->get(object => "Infoblox::Grid::Admin::LocalUserAuthService");

#Set it as the authentication policy
$auth_policy->auth_services([$local]);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a Local User Authentication Service object.

Infoblox::Session->get( )

Use this method to retrieve the local user authentication service from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

No parameters are required.

Example

```
my $local_user_auth_service = $session->get(
    object => "Infoblox::Grid::Admin::LocalUserAuthService",
    );
```

METHODS

This section describes all the methods that you can use to retrieve the attribute values of a local user authentication service.

comment( )

Use this method to retrieve the descriptive comment.

Returns

The method returns the attribute value.

Example

```
#Get comment
my $comment = $local_user_auth_service->comment();
```

disabled( )

Use this method to retrieve the disabled flag for the local user authentication service.

Returns

The method returns the attribute value.

Example

```
#Get disabled
my $disabled = $local_user_auth_service->disabled();
```

name( )

Use this method to retrieve the name of the local user authentication service.

Returns

The method returns the attribute value.

Example

```
#Get name
my $name = $local_user_auth_service->name();
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Admin::AuthPolicy/>, Infoblox::Session->get()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::Permission - Admin Permission object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - admin_group( )
  - role( )
  - permission( )
  - resource_type( )
  - resource_object( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::Permission - Admin Permission object.

DESCRIPTION

An Admin Permission object is used to set the permissions for an admin to access various resources.

CONSTRUCTOR

```
 my $group = Infoblox::Grid::Admin::Permission->new(
     admin_group         => $string,                                                    #exactly
one of admin_group and role Required
     role                => $string,                                                    #exactly
one of admin_group and role Required
     permission          => "read" | "write" | "deny",                                   #Required
     resource_type       => $string,                                                    #Optional,
if resource_object is provided
     resource_object     => $Object,                                                    #Optional,
if resource_type is provided
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to an Admin Permission object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the Group object
my $permission = Infoblox::Grid::Admin::Permission->new(
          admin_group => "test_group",
          resource_type => "All Zones",
          permission => "write",
);
# Submit for addition
my $response = $session->add( $permission );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific Group object:

 admin_group      - Optional. Group name in string format.
 role             - Optional. Role name in string format.
 permission       - Optional. Permission in string format.
 resource_type    - Optional. Resource type in string format.
 resource_object  - Optional. Resource Object reference.

NOTE: You cannot specify both an admin_group and role.
```

Example

```
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::Permission",
    admin_group   => "test_group",
    resource_type => "All Members");

or

my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::Permission",
    role          => "test_role",
    resource_type => "All Members");
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the permission.
$permission->permission("read");
# Submit modification
my $response = $session->modify( $permission );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the permission objects with the same group name
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::Permission",
    admin_group   => "test_group",
    resource_type => "All Members");
# find the desired object from the retrieved list.
my $desired_perm = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_perm );

or

# Get the permission objects with the same role name
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::Permission",
    role          => "test_role",
    resource_type => "All Members");
# find the desired object from the retrieved list.
my $desired_perm = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_perm );
```

Infoblox::Session->search( )

Use this method to search for Admin Permission objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific Group object:

 admin_group          - Optional. Group name in string format.
 role                 - Optional. Role name in string format.
 permission           - Optional. Permission in string format.
 resource_object      - Optional. Resource Object reference.
 resource_type        - Optional. Resource Type in string format.
```

NOTE: You cannot specify both an admin_group and role.

Example

```
# search for all permission objects of admin group "test_group"
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::Admin::Permission",
    admin_group   => "test_group" );
```

or

```
# search for all permission objects of admin role "test_role"
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::Admin::Permission",
    role   => "test_role" );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a Group object.

admin_group( )

Use this method to set or retrieve the admin group information.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The admin group name for the specified object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get admin group name
my $name = $permission->admin_group();
#Modify admin group
$permission->admin_group("test-group1");
```

role( )

Use this method to set or retrieve the role information.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The admin role name for the specified object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get admin role name
my $name = $permission->role();
#Modify admin role
$permission->role("test-role1");
```

permission( )

Use this method to set or retrieve the admin group permission.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The admin group permission for the specified object. Permission values are "write", "read" and "deny".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get permission
my $perm = $permission->permission();
#Modify permission
$permission->permission("deny");
```

resource_type( )

Use this method to set or retrieve the resource type. Resource type represents all resources of the specific type.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The resource type for which permission is set. Sets permissions for all resources of the resource_type specified.

If resource_object is also provided, it sets the permission for all resources of the specified type under the resource_object.

The allowed values are:

```
"All Zones"
"All DNS Views"
"All DNS64 Synthesis Groups"
"All Members"
"All MS Server Members"
"All Microsoft Superscopes"
"All Shared Record Groups"
"All Networks" (deprecated in favor of "All IPv4 Networks")
"All IPv4 Networks"
"All IPv6 Networks"
"All Network Views"
"All Shared Networks" (deprecated in favor of "All IPv4 DHCP Shared Networks")
"All IPv4 DHCP Shared Networks"
"All IPv6 DHCP Shared Networks"
"All Mac Address Filters"
"All Ranges" (deprecated in favor of "All IPv4 DHCP Ranges")
"All IPv4 DHCP Ranges"
"All IPv6 DHCP Ranges"
"All Fixed Addresses/Reservations" (deprecated in favor of "All IPv4 DHCP Fixed
Addresses/Reservations")
"All IPv4 DHCP Fixed Addresses/Reservations"
"All IPv6 DHCP Fixed Addresses"
"All Roaming Hosts"
"All Rulesets"
"All Templates" (deprecated in favor of "All IPv4 DHCP Templates")
"All IPv4 DHCP Templates"
"All IPv6 DHCP Templates"
"All A Records"
"All AAAA Records"
"All BULKHOST Records"
"All CNAME Records"
"All DNAME Records"
"All HOST Records"
"All MX Records"
"All NS Records"
"All PTR Records"
"All SRV Records"
"All TXT Records"
"All Shared A Records"
"All Shared AAAA Records"
"All Shared MX Records"
"All Shared SRV Records"
"All Shared TXT Records"
"Lease History Access" (deprecated in favor of "All IPv4 DHCP Lease History")
"All IPv4 DHCP Lease History"
"All IPv6 DHCP Lease History"
"Grid File Distribution Properties"
"Grid AAA Properties"
"All AAA Local Users"
"All AAA Replicated Users"
"All AAA External Services"
"All AAA Policies"
"Network Discovery"
"Scheduling"
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get resource_type
```

```
my $resource_type = $permission->resource_type();
#Modify resource_type
$permission->resource_type("All Networks");
```

resource_object( )

Use this method to set or retrieve a resource object. Resource object is a reference to a single object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The resource object for which permission is set. Sets the permission for the specified resource object only.

If resource_type is also provided, it sets the permission for all resources of the specified type under the resource_object.

The value must be a reference of a defined object of any of the following types:

```
Infoblox::DHCP::Filter::MAC
Infoblox::DHCP::FixedAddr
Infoblox::DHCP::FixedAddrTemplate
Infoblox::DHCP::IPv6FixedAddr
Infoblox::DHCP::IPv6Network
Infoblox::DHCP::IPv6NetworkContainer
Infoblox::DHCP::IPv6Range
Infoblox::DHCP::IPv6SharedNetwork
Infoblox::DHCP::Network
Infoblox::DHCP::NetworkContainer
Infoblox::DHCP::NetworkTemplate
Infoblox::DHCP::Range
Infoblox::DHCP::RangeTemplate
Infoblox::DHCP::RoamingHost
Infoblox::DHCP::SharedNetwork
Infoblox::DHCP::View
Infoblox::DNS::BulkHost
Infoblox::DNS::Host
Infoblox::DNS::Record::A
Infoblox::DNS::Record::AAAA
Infoblox::DNS::Record::CNAME
Infoblox::DNS::Record::DNAME
Infoblox::DNS::Record::MX
Infoblox::DNS::Record::PTR
Infoblox::DNS::Record::SRV
Infoblox::DNS::Record::TXT
Infoblox::DNS::Ruleset
Infoblox::DNS::SharedRecord::A
Infoblox::DNS::SharedRecord::AAAA
Infoblox::DNS::SharedRecord::MX
Infoblox::DNS::SharedRecord::SRV
Infoblox::DNS::SharedRecord::TXT
Infoblox::DNS::SRG
Infoblox::DNS::View
Infoblox::DNS::Zone
Infoblox::Grid::DNS::DNS64Group
Infoblox::Grid::FileDistributionDir
Infoblox::Grid::Member
Infoblox::Grid::Member::FileDistribution
Infoblox::Grid::MSServer
Infoblox::Grid::MSSuperscope
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get resource_obejct
my $resource_object = $permission->resource_object();
#Modify resource_object
$zone = Infoblox::DNS::Zone->new(
               name => "zone.com"
               );
$session->add($zone);
$permission->resource_object($zone);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to an Admin Permission object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
```

```
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create an admin group

my $group = Infoblox::Grid::Admin::Group->new(
    name                 => "test_group",
    disabled             => "false",
    superuser            => "false"

);

unless ($group) {
       die("Construct user object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Admin Group object created successfully\n";

#Add the Admin Group object to the Infoblox appliance through a session
$session->add($group)
       or die("Add Admin Group object failed: ",
                     $session->status_code() . ":" . $session->status_detail());
print "Admin Group object added to server successfully\n";

# Create an Admin Role

my $role = Infoblox::Grid::Admin::Role->new(
    name                 => "test_role",
    comment              => "test admin role",
);

unless ($role) {
       die("Construct role object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Admin Role object created successfully\n";

#Add the Admin Role object to the Infoblox appliance through a session
$session->add($role)
       or die("Add Admin Role object failed: ",
                     $session->status_code() . ":" . $session->status_detail());
print "Admin Role object added to server successfully\n";

#Create a zone
my $zone = Infoblox::DNS::Zone->new(
              name => "zone.com"
);

unless ($zone) {
       die("Construct user object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}

#Add the Zone object to the Infoblox appliance through a session
$session->add($zone)
       or die("Add Zone object failed: ",
                     $session->status_code() . ":" . $session->status_detail());
print "Zone object added to server successfully\n";

#Create a shared record group (SRG)
my $srg = Infoblox::DNS::SRG->new(
              name => "mysrg"
);
```

```
unless ($srg) {
       die("Construct user object failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}

#Add the SRG object to the Infoblox appliance through a session
$session->add($srg)
       or die("Add SRG object failed: ",
                       $session->status_code() . ":" . $session->status_detail());
print "SRG object added to server successfully\n";
```

#Create an Admin Permission object based on Admin Group

```
#Create an admin permission object
my $group_permission = Infoblox::Grid::Admin::Permission->new(
   admin_group     => $group->name(),
   resource_object => $zone,  # This can be omitted if don't care about
                              # which zone
   resource_type => "All A Records",
   permission => "write");

unless ($group_permission) {
       die("Construct Admin Permission object failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Admin Permission object created successfully\n";

#Add the Admin Permission object to the Infoblox appliance through a session
$session->add($group_permission)
       or die("Add Admin Permission object failed: ",
                       $session->status_code() . ":" . $session->status_detail());
print "Admin Permission object added to server successfully\n";

#Create an admin permission object
my $scheduling_permission = Infoblox::Grid::Admin::Permission->new(
     admin_group => $group->name(),
     resource_type => "Scheduling",
     permission => "write");

unless ($scheduling_permission) {
       die("Construct Admin Permission object failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Admin Permission object created successfully\n";

#Add the Admin Permission object to the Infoblox appliance through a session
$session->add($scheduling_permission)
       or die("Add Admin Permission object failed: ",
                       $session->status_code() . ":" . $session->status_detail());
print "Admin Permission object added to server successfully\n";
```

#Create an Admin Permission object based on Admin Role

```
#Create an admin permission object
my $role_permission = Infoblox::Grid::Admin::Permission->new(
   role            => $role->name(),
   resource_object => $srg,  # This can be omitted if don't care about
                             # which shared record group
   resource_type => "All Shared A Records",
   permission => "write");

unless ($role_permission) {
       die("Construct user object failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Admin Permission object created successfully\n";

#Add the Admin Permission object to the Infoblox appliance through a session
$session->add($role_permission)
       or die("Add Admin Permission object failed: ",
                       $session->status_code() . ":" . $session->status_detail());
print "Admin Permission object added to server successfully\n";
```

#Search for an Admin Permission object based on Admin Group

```
my @retrieved_objs = $session->search(
               object => "Infoblox::Grid::Admin::Permission",
```

```
                admin_group   => "test_group"
                );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search Admin Permission object based on Admin Group failed: ",
                $session->status_code() . ":" . $session->status_detail());
}
print "Search Admin Permission object based on Admin Group found at least 1 matching entry\n";
```

#Search for an Admin Permission object based on Admin Role

```
my @retrieved_objs = $session->search(
                object => "Infoblox::Grid::Admin::Permission",
                role   => "test_role"
                );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search Admin Permission object based on Admin Role failed: ",
                $session->status_code() . ":" . $session->status_detail());
}
print "Search Admin Permission object based on Admin Role found at least 1 matching entry\n";
```

#Get and modify an Admin Permission object based on Admin Group

```
#Get the Admin Group object from the Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::Permission",
    admin_group   => "test_group",
    resource_object => $zone,
    resource_type => "All A Records",
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get Admin Permission object based on Admin Group failed: ",
                $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin Permission object found at least 1 matching entry\n";

#Modify the permission of the Admin Permission object
$object->permission("read");

#Apply the changes.
$session->modify($object)
       or die("Modify Admin Permission object failed: ",
                $session->status_code() . ":" . $session->status_detail());
print "Admin Permission object modified successfully \n";
```

#Get Admin Permission object based on Admin Role

```
#Get the Admin Group object from the Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::Permission",
    role           => "test_role",
    resource_object => $srg,
    resource_type => "All Shared A Records",
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get Admin Permission object based on Admin Role failed: ",
                $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin Permission object found at least 1 matching entry\n";
```

#Remove an Admin Permission object

```
#Get the Admin Permission object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::Permission",
    admin_group   => "test_group",
    resource_object => $zone,
    resource_type => "All A Records",
);
my $object = $retrieved_objs[0];
```

```
unless ($object) {
    die("Get Admin Permission object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin Permission object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove Admin Permission object failed: ",
                $session->status_code() . ":" . $session->status_detail());
print "Admin Permission object removed successfully \n";

# We can remove the other permissions in the same way

#Get the Admin Permission object for Shared Resource Records through the session
@retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::Permission",
    role   => "test_role",
    resource_object => $srg,   # Omit this if it was omitted during creation
    resource_type => "All Shared A Records",
);
$object = $retrieved_objs[0];
unless ($object) {
    die("Get Admin Permission object for Shared RR failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin Permission object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove Admin Permission object for Shared RR failed: ",
                $session->status_code() . ":" . $session->status_detail());
print "Admin Permission object for Shared RR removed successfully \n";

#Clean up

$session->remove($role)
       or die("Remove Admin Role object failed: ",
                $session->status_code() . ":" . $session->status_detail());
print "Admin Role object removed successfully \n";

$session->remove($group)
       or die("Remove Admin Group object failed: ",
                $session->status_code() . ":" . $session->status_detail());
print "Admin Group object removed successfully \n";

$session->remove($zone)
       or die("Remove Zone object failed: ",
                $session->status_code() . ":" . $session->status_detail());
print "Zone object removed successfully \n";

$session->remove($srg)
       or die("Remove Shared Record Group object failed: ",
                $session->status_code() . ":" . $session->status_detail());
print "Shared Record Group object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Admin::Role, Infoblox::Grid::Admin::Group, Infoblox::Grid::Admin::User, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::RadiusAuthServer - RADIUS Authentication Server object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Grid::Admin::RadiusAuthService->radius_servers()
- METHODS
  - acct_enabled( )
  - acct_port( )
  - auth_port( )
  - auth_type( )
  - comment( )
  - disabled( )
  - fqdn_or_ip( )
  - shared_secret( )
  - use_mgmt_port( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::RadiusAuthServer - RADIUS Authentication Server object

DESCRIPTION

Represents a RADIUS server that is used to authenticate administrators and DHCP clients.

CONSTRUCTOR

```
my $radius_auth_server = Infoblox::Grid::Admin::RadiusAuthServer->new(
    acct_enabled     => "true" | "false"      #Optional / Default is false
    acct_port        => $num,                 #Optional / Default is 1813
    auth_port        => $num,                 #Optional / Default is 1812
    auth_type        => "PAP" | "CHAP",       #Optional / Default is "PAP"
    comment          => $string,              #Optional / Default is empty
    disabled         => "true" | "false"      #Optional / Default is false
    fqdn_or_ip       => $fqdn | $ipv4addr     #Required
    shared_secret    => $string,              #Required
    use_mgmt_port    => "true" | "false"      #Optional / Default is false
);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to an object.

Infoblox::Grid::Admin::RadiusAuthService->radius_servers()

Use this method to add a list of RADIUS Server objects to the RADIUS Service object on an Infoblox appliance. See Infoblox::Grid::Admin::RadiusAuthService for parameters and return values.

Example

```
# Construct an object
my $radius_auth_server1 = Infoblox::Grid::Admin::RadiusAuthServer->new(
                                                        fqdn_or_ip    => '10.0.1.1',
                                                        shared_secret => 'secret',
);

# Construct another object
my $radius_auth_server2 = Infoblox::Grid::Admin::RadiusAuthServer->new(
                                                        fqdn_or_ip    => '10.0.2.1',
                                                        shared_secret => 'secret',
);

# Configure RadiusAuthServer
my $response = $radius_auth_service->radius_servers([$radius_auth_server1, $radius_auth_server2]);
```

METHODS

Use the following methods to access the attributes of an Infoblox::Grid::Admin::RadiusAuthServer object. Specify a parameter to set the attribute's value, or omit the parameter to get the attribute's value.

acct_enabled( )

Use this method to set or retrieve the flag that indicates if the RADIUS server is enabled for accounting packets.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get acct_enabled
my $acct_enabled = $radius_server->acct_enabled();
#Modify acct_enabled
$radius_server->acct_enabled("true");
```

acct_port( )

Use this method to set or retrieve the port number used by RADIUS server for accounting packets.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 1 and 65535, inclusive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting acct_port
my $acct_port = $radius_server->acct_port( );
#Modifying acct_port
$radius_server->acct_port("1813");
```

auth_port( )

Use this method to set or retrieve the port number used by the RADIUS server for accounting packets.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 1 and 65535, inclusive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting auth_port
my $auth_port = $radius_server->auth_port( );
#Modifying auth_port
$radius_server->auth_port("1812");
```

auth_type( )

Use this method to set or retrieve the authentication type used by the RADIUS client and server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is either "PAP" or "CHAP", the default is "PAP".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting auth_type
my $auth_type = $radius_server->auth_type( );
#Modifying auth_type
$radius_server->auth_type("PAP");
```

comment( )

Use this method to set or retrieve a descriptive comment about the RADIUS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The comment in string format, with a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $radius_server->comment();
#Modify comment
$radius_server->comment("NAC Authentication RADIUS server");
```

disabled( )

Use this method to set or retrieve the disable flag of the RADIUS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disabled
my $disable = $radius_server->disabled();
#Modify disabled
$radius_server->disabled("true");
```

fqdn_or_ip( )

Use this method to set or retrieve the FQDN or IP address of the RADIUS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A string in FQDN (Fully Qualified Domain Name) format, or an IP address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get fqdn_or_ip
my $fqdn_or_ip = $radius_server->fqdn_or_ip();
#Modify fqdn_or_ip
$radius_server->fqdn_or_ip("10.0.0.1");
```

shared_secret( )

Use this method to set or retrieve the shared secret used by the RADIUS client and RADIUS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The shared secret in string format, with a maximum of 128 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get shared_secret
my $shared_secret = $radius_server->shared_secret();
#Modify shared_secret
$radius_server->shared_secret("secret");
```

use_mgmt_port( )

Use this method to set or retrieve the flag that indicates if the appliance connects to the RADIUS server through the MGMT network interface.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get use_mgmt_port
my $use_mgmt_port = $radius_server->use_mgmt_port();
#Modify use_mgmt_port
$radius_server->use_mgmt_port("true");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com

SEE ALSO

Infoblox::Grid::Admin::RadiusAuthService, Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::RadiusAuthService - RADIUS Authentication Service object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
- METHODS
  - acct_retries( )
  - acct_timeout( )
  - auth_retries( )
  - auth_timeout( )
  - cache_ttl( )
  - comment( )
  - disabled( )
  - enable_cache( )
  - name( )
  - selection_mode( )
  - radius_servers( )
  - recovery_interval( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::RadiusAuthService - RADIUS Authentication Service object

DESCRIPTION

Represents a group of RADIUS servers that are used to authenticate administrators and DHCP clients.

CONSTRUCTOR

```
my $radius_auth_server = Infoblox::Grid::Admin::RadiusAuthService->new(
    acct_retries          => $num,                                  #Optional / Default is 1000
    acct_timeout          => $num,                                  #Required
    auth_retries          => $num,                                  #Optional / Default is 5
    auth_timeout          => $num,                                  #Required
    cache_ttl             => $num,                                  #Optional / Default is 3600
    comment               => $string,                               #Optional / Default is empty
    disabled              => "true" | "false"                       #Optional / Default is false
    enable_cache          => "true" | "false"                       #Optional / Default is false
    name                  => $string,                               #Required
    radius_servers        => [$RadiusAuthServer1, ...],              #Required
    recovery_ttl          => $num,                                  #Optional / Default is 30
    selection_mode        => "HUNT_GROUP" | "ROUND_ROBIN",           #Optional / Default is "HUNT_GROUP"
);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a RADIUS Auth object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $service = Infoblox::Grid::Admin::RadiusAuthService->new(
                                                            name           => 'some.name.com',
                                                            radius_servers => [$ads1, $ads2]
                                                           );
```

```
#Submit for addition
my $response = $session->add( $service );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific RADIUS Auth object:

 name - Required. RADIUS Authentication Service name.
```

Examples

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::RadiusAuthService",
    name   => "some.name.com"
);
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use this method to modify the comment.
$service->comment("this is a modified comment");
#Submit modification
my $response = $session->modify( $service );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::RadiusAuthService",
    name   => "some.name.com"
);

#Find the desired object from the retrieved list.
my $desired_service = $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_service );
```

METHODS

Use the following methods to access the attributes of an Infoblox::Grid::Admin::RadiusAuthService object. Specify a parameter to set the attribute's value, or omit the parameter to get the attribute's value.

acct_retries( )

Use this method to set or retrieve the value that specifies how many times the appliance must resend the RADIUS accounting packet if the appliance does not receive a response.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 1 and 10, inclusive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting acct_retries
my $acct_retries = $radius_service->acct_retries( );
```

```
#Modifying acct_retries
$radius_service->acct_retries("5");
```

acct_timeout( )

Use this method to set or retrieve the value, in milliseconds, that specifies how long the appliance must wait for authentication response packets.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 1000 and 10000, inclusive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting acct_timeout
my $acct_timeout = $radius_service->acct_timeout( );
#Modifying acct_timeout
$radius_service->acct_timeout("5000");
```

auth_retries( )

Use this method to set or retrieve the value of how many times to retry if there is no response for the RADIUS authentication packet.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 1 and 10, inclusive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting auth_retries
my $auth_retries = $radius_service->auth_retries( );
#Modifying auth_retries
$radius_service->auth_retries("5");
```

auth_timeout( )

Use this method to set or retrieve the value in milliseconds to wait for authentication response packets.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 1000 and 10000, inclusive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting auth_timeout
my $auth_timeout = $radius_service->auth_timeout( );
#Modifying auth_timeout
$radius_service->auth_timeout("5000");
```

cache_ttl( )

Use this method to set or retrieve the TTL for the authentication response in the authentication cache.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 1 and 259200, inclusive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting cache_ttl
my $cache_ttl = $radius_service->cache_ttl( );
#Modifying cache_ttl
$radius_service->cache_ttl("3600");
```

comment( )

Use this method to set or retrieve a descriptive comment about the RADIUS Authentication Service object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The comment in string format, with a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $radius_service->comment( );
#Modify comment
$radius_service->comment("RADIUS Authentication Service for NAC authentication");
```

disabled( )

Use this method to set or retrieve the disable flag of the RADIUS Authentication Service object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $radius_service->disabled( );
#Modify disabled
$radius_service->disabled("true");
```

enable_cache( )

Use this method to set or retrieve the flag that indicates if the authentication cache is enabled on the RADIUS Authentication Service object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_cache
my $enable_cache = $radius_service->enable_cache( );
#Modify enable_cache
$radius_service->enable_cache("true");
```

name( )

Use this method to set or retrieve the name of the RADIUS Authentication Service object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name in string format, with a maximum of 64 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting name
my $name = $radius_service->name( );
#Modifying name
$radius_service->name("RADIUS Authentication Service");
```

selection_mode( )

Use this method to set or retrieve the process the appliance uses to select to which RADIUS server in the RADIUS Authentication Service it sends RADIUS requests.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "HUNT_GROUP" (ordered list) or "ROUND_ROBIN". The default is "HUNT_GROUP".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting selection_mode
my $selection_mode = $radius_service->selection_mode( );
#Modifying selection_mode
$radius_service->selection_mode("HUNT_GROUP");
```

radius_servers( )

Use this method to set or retrieve a list of RADIUS servers assigned to this RADIUS Service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An array reference that contains a list of Infoblox::Grid::Admin::RadiusAuthServer objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting radius_servers
my $radius_servers = $radius_service->radius_servers( );
#Modifying radius_servers
$radius_service->radius_servers([$radius_server1]);
```

recovery_interval( )

Use this method to set or retrieve the time in seconds that the appliance waits before re-enabling a RADIUS server marked as DOWN.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An unsigned integer between 1 and 600, inclusive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting recovery_interval
my $recovery_interval = $radius_service->recovery_interval( );
#Modifying recovery_interval
$radius_service->recovery_interval("30");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a Radius Auth Service object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $host_ip =  "192.168.1.2";
my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
    );
unless($session){
        die("Constructor for session failed: ",
               Infoblox::status_code(). ":" . Infoblox::status_detail());
}
print "Session created successfully.\n";

my $ads1 = Infoblox::Grid::Admin::RadiusAuthServer->new(
                                                        comment  => 'server 1',
                                                        fqdn_or_ip => '10.0.1.1',
                                                        shared_secret => 'secret1',
                                                       );

my $ads2 = Infoblox::Grid::Admin::RadiusAuthServer->new(
                                                        comment  => 'server 2',
                                                        fqdn_or_ip => 'domain.com',
                                                        shared_secret => 'secret2',
                                                      );
```

#Create the RADIUS Auth Service.

```
my $service = Infoblox::Grid::Admin::RadiusAuthService->new(
                                                            acct_retries   => 2000,
                                                            auth_retries   => 9,
                                                            cache_ttl      => 500,
                                                            disabled       => 'true',
                                                            enable_cache   => 'true',
                                                            recovery_ttl   => 50,
                                                            selection_mode => 'ROUND_ROBIN',
                                                            acct_timeout   => 1100,
                                                            auth_timeout   => 1200,
                                                            name           => 'some.name.com',
                                                            radius_servers => [$ads1, $ads2]
                                                          );

unless($service){
       die("Construct service object failed: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
```

```
        }
 print "test.com zone object created successfully.\n";

 $session->add($service)
     or die("Add zone service failed: ",
                $session->status_code(). ":" .$session->status_detail());

 print"Service added successfully.\n";

#Get a specific service

 my @search_result = $session->get(
     object => "Infoblox::Grid::Admin::RadiusAuthService",
     name   => "some.name.com"
 );

 my $search_obj = $search_result[0];
 unless($search_obj){
        die("Search service failed: ",
                 $session->status_code() . ":" . $session->status_detail());
        }

 print "Search service object found at least 1 matching entry\n";

#Get and modify a RADIUS service

 $search_obj->comment("this is a modified comment.");

 #Apply the change
     $session->modify($search_obj)
         or die("Modify service failed: ",
                $session->status_code(). ":" .$session->status_detail());

  print "Service object modified successfully.\n";

#Remove a Service

 my @search_result = $session->get(
     object => "Infoblox::Grid::Admin::RadiusAuthService",
     name   => "some.name.com"
 );

 my $search_obj = $search_result[0];
 unless($search_obj){
        die("Search service failed: ",
                 $session->status_code() . ":" . $session->status_detail());
        }

 print "Search service object found at least 1 matching entry\n";

 $session->remove($search_obj)
     or die("Remove service failed: ",
            $session->status_code(). ":" .$session->status_detail());

 print "Remove service successfull\n";

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com

SEE ALSO

Infoblox::Grid::Admin::RadiusAuthServer, Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::Role - Admin Role object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - disabled( )
  - extensible_attributes( )
  - name( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::Role - Admin Role object.

DESCRIPTION

An Admin Role object creates and manages a local admin role on the Infoblox appliance. A Role object is used to aggregate a set of permissions (represented by Permission objects).

CONSTRUCTOR

```
 my $role = Infoblox::Grid::Admin::Role->new(
     name                  => $string,
#Required
     comment               => $string,
#Optional / Default is undefined
     disabled              => "true" | "false",
#Optional / Default is "false"
     extensible_attributes => { $string => $string | $num, $string => [ $string | $num, ... ], ... }
#Optional / Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session module that you can apply to an Admin Role object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the Role object
my $role = Infoblox::Grid::Admin::Role->new(
    name                  => "testrole",
    comment               => "test role",
);
# Submit for addition
my $response = $session->add( $role );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific Role object:
```

```
 name      - Required. Role name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::Role",
    name       => "testrole" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment.
$role->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $role );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the role objects with the same starting name
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::Admin::Role",
    name       => "testrole" );
# find the desired object from the retrieved list.
my $desired_role = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_role );
```

Infoblox::Session->search( )

Use this method to search for Admin Role objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific Role object:

 name      - Required. Role name in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all Role objects that start with "test"
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::Admin::Role",
    name   => "test.*" );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a Role object.

comment( )

Use this method to set or retrieve the comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Pertinent information about the admin role, such as its purpose. Comment in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $role->comment();
#Modify comment
$role->comment("This is the modified comment for admin role");
```

disabled( )

Use this method to set or retrieve the "disabled" flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The role is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disabled
my $disabled = $role->disabled();
#Modify disabled
$role->disabled("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a Role object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $role->extensible_attributes();
#Modify extensible attributes
$role->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the name of an admin role.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name of the admin role with a maximum of 64 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $role->name();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to an Admin Role object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create an Admin Role object

```
my $role = Infoblox::Grid::Admin::Role->new(
    name                 => "testrole",
    comment              => "test admin role",
);

unless ($role) {
       die("Construct role object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Admin Role object created successfully\n";

#Add the Admin Role object to the Infoblox appliance through a session
$session->add($role)
       or die("Add Admin Role object failed: ",
                     $session->status_code() . ":" . $session->status_detail());
print "Admin Role object added to server successfully\n";
```

#Search for an Admin Role

```
my @retrieved_objs = $session->search(
              object => "Infoblox::Grid::Admin::Role",
              name   => "test.*"
              );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search Admin Role object failed: ",
              $session->status_code() . ":" . $session->status_detail());
}
print "Search Admin Role object found at least 1 matching entry\n";
```

#Get and modify an Admin Role object

```
#Get the Admin Role object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::Role",
    name   => "testrole"
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get Admin Role object failed: ",
              $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin Role object found at least 1 matching entry\n";

#Modify the Admin Role object
$object->name("testrole_renamed");
```

```
$object->comment("this is a new comment");

#Apply the changes.
$session->modify($object)
       or die("Modify Admin Role object failed: ",
              $session->status_code() . ":" . $session->status_detail());
print "Admin Role object modified successfully \n";

#Accessor methods
print "name: " . $object->name() . "\n";
print "comment: " . $object->comment() . "\n";
```

#Remove an Admin Role object

```
#Get the Admin Role object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Admin::Role",
    name   => "testrole_renamed"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Admin Role object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin Role object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove Admin Role object failed: ",
              $session->status_code() . ":" . $session->status_detail());
print "Admin Role object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Admin::Group, Infoblox::Grid::Admin::Permission, Infoblox::Grid::Admin::User, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Admin::User - User Admin object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - admin_group( )
  - comment( )
  - disabled( )
  - email( )
  - extensible_attributes( )
  - name( )
  - password( )
  - time_zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Admin::User - User Admin object.

DESCRIPTION

An admin account provides access to the Infoblox appliance. An admin account inherits the privileges and properties of the group to which it belongs.

CONSTRUCTOR

```
 my $user = Infoblox::Grid::Admin::User->new(
     admin_group            => $string,            #Required
     name                   => $string,            #Required
     password               => $string,            #Required
     comment                => $string,            #Optional / Default is undefined
     disabled               => "true" | "false",   #Optional / Default is "false"
     extensible_attributes  => { $string => $string | $num, $string => [ $string | $num, ... ], ... }
#Optional / Default is undefined
     email                  => $email,             #Optional / Default is undefined
     time_zone              => $TimeZone | undef   #Optional / Default value is undefined
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to an Admin User object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the user object
my $user = Infoblox::Grid::Admin::User->new(
    admin_group => "admin-group",
    name        => "testuser",
    password    => "infoblox",
    comment     => "test user account"
    disabled    => "true",
    email       => "testuser\@test.com",
);
# Submit for addition
my $response = $session->add( $user );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific User object:

 name           - Required. User name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object      => "Infoblox::Grid::Admin::User",
    name        => "testuser" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the comment.
$user->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $user );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the user objects with the same starting name
my @retrieved_objs = $session->get(
    object      => "Infoblox::Grid::Admin::User",
    name        => "testuser" );
# find the desired object from the retrieved list.
my $desired_user = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_user );
```

Infoblox::Session->search( )

Use this method to search for Admin User objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific User object:

 name           - Required. User name in string format (regular expression).
 admin_group    - Optional. Admin group name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all User objects that start with "test"
my @retrieved_objs = $session->search(
    object      => "Infoblox::Grid::Admin::User",
    name        => "test.*",
    admin_group => "admin-group" );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a User object.

admin_group( )

Use this method to set or retrieve the admin_group.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Admin group to which the user belongs. A user can belong to only one admin group at a time.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get admin group
my $admin_group = $user->admin_group();
#Modify name
$user->name("admin-group");
```

comment( )

Use this method to set or retrieve a comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Pertinent information about the administrator, such as location or department. Comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $user->comment();
#Modify comment
$user->comment("This is the modified comment for admin user");
```

disabled( )

Use this method to set or retrieve the "disabled" flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value for this field is false. The admin is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disabled
my $disabled = $user->disabled();
#Modify disabled
$user->disabled("true");
```

email( )

Use this method to set or retrieve the e-mail address of an administrator.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The e-mail address of an administrator. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get email
my $email = $user->email();
#Modify email
$user->email("usertest\@infoblox.com");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a User object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $user->extensible_attributes();
#Modify extensible attributes
$user->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the name of an administrator.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name of the admin.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $user->name();
#Modify name
$user->name("test_user");
```

password( )

Use this method to set or retrieve the password of an administrator.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Password for the administrator to use when logging in.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the ssha encrypted attribute value.

Example

```
#Get password
my $password = $user->password();
#Modify password
$user->password("infobloxone");
```

time_zone( )

Use this method to set or retrieve the time zone of the administrator.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The UTC string that represents the time zone. For example "(UTC - 6:00) Central Time (US and Canada)". Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get time zone
my $time_zone = $user->time_zone();
#Modify time zone
$user->time_zone("(UTC - 6:00) Central Time (US and Canada)");
#Undef time zone
$user->time_zone(undef);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to an Admin User object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
               );

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create an Admin User object

```
my $user = Infoblox::Grid::Admin::User->new(
    name        => "testuser",
    password    => "infoblox",
    admin_group => "admin-group",
    email       => "testuser\@test.com",
    disabled    => "false",
    comment     => "test user account"
);

unless ($user) {
       die("Construct user object failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "User object created successfully\n";

#Add the user object to the Infoblox appliance through a session
$session->add($user)
       or die("Add User object failed: ",
                       $session->status_code() . ":" . $session->status_detail());
```

```
print "Admin User object added to server successfully\n";
```

#Search for all users

```
my @retrieved_objs = $session->search(
               object      => "Infoblox::Grid::Admin::User",
               name        => "test.*",
               admin_group => "admin-group"
              );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search User object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search Admin User object found at least 1 matching entry\n";
```

#Get and modify a User object

```
#Get the user object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
            object      => "Infoblox::Grid::Admin::User",
            name        => "testuser",
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get User object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Get Admin User object found at least 1 matching entry\n";

#Modify the password of the User object
$object->password("infoblox123");

#Apply the changes.
$session->modify($object)
       or die("Modify User object failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "User object modified successfully \n";
```

#Remove the User object

```
#Get the User object through the session
my @retrieved_objs = $session->get(
            object      => "Infoblox::Grid::Admin::User",
            name        => "testuser",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get User object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get User object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove User object failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "Admin User object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Admin::Group, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::CSVImportStatus - CSV import status object

- NAME
- DESCRIPTION
- MODULE METHODS
  - Infoblox::Session->get( )
- METHODS
  - admin( )
  - end_time( )
  - file_name( )
  - import_id( )
  - lines_failed( )
  - lines_processed( )
  - lines_total( )
  - lines_warning( )
  - start_time( )
  - status( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::CSVImportStatus - CSV import status object

DESCRIPTION

This object is used to represent the status of the current CSV import task. All methods are read-only.

MODULE METHODS

The following functions can be applied to a Grid::CSVImportStatus object.

Infoblox::Session->get( )

Use this method to retrieve matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get a specific CSV import status object

 object - Required. It must be set to "Infoblox::Grid::CSVImportStatus".
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::CSVImportStatus"
    );
```

METHODS

This section describes all the methods that you can use to retrieve the status of a CSV import operation.

admin( )

Use this method to retrieve the name of the administrator that started the import.

Returns

The method returns the attribute value.

Example

```
#Get admin
my $admin = $service_status->admin();
```

end_time( )

Use this method to retrieve the time the CSV import ended.

Returns

The method returns the attribute value.

Example

```
#Get end_time
my $end_time = $service_status->end_time();
```

file_name( )

Use this method to retrieve the name of the file used for the CSV import.

Returns

The method returns the attribute value.

Example

```
#Get file_name
my $file_name = $service_status->file_name();
```

import_id( )

Use this method to retrieve the ID of the import task that the status object is describing.

Returns

The method returns the attribute value.

Example

```
#Get import_id
my $import_id = $service_status->import_id();
```

lines_failed( )

Use this method to retrieve the number of lines that had failures in the CSV file used for the current CSV import.

Returns

The method returns the attribute value.

Example

```
#Get lines_failed
my $lines_failed = $service_status->lines_failed();
```

lines_processed( )

Use this method to retrieve the number of lines processed in the CSV file used for the current CSV import.

Returns

The method returns the attribute value.

Example

```
#Get lines_processed
my $lines_processed = $service_status->lines_processed();
```

lines_total( )

Use this method to retrieve the total number of lines read from the CSV file during the current CSV import.

Returns

The method returns the attribute value.

Example

```
#Get lines_total
my $lines_total = $service_status->lines_total();
```

lines_warning( )

Use this method to retrieve the number of lines that had warnings in the CSV file used for the current CSV import.

Returns

The method returns the attribute value.

Example

```
#Get lines_warning
my $lines_warning = $service_status->lines_warning();
```

start_time( )

Use this method to retrieve the time the CSV import started.

Returns

The method returns the attribute value.

Example

```
#Get start_time
my $start_time = $service_status->start_time();
```

status( )

Use this method to retrieve the status of the current CSV import. The returned status is one of the following: 'Uploaded', 'Pending', 'Running', 'Failed', 'Stopped' or 'Complete'.

Returns

The method returns the attribute value.

Example

```
#Get status
my $status = $service_status->status();
```

SAMPLE CODE

The following sample code demonstrates how to export / import data by using CSV files.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $host_ip =  "192.168.1.2";
my $session = Infoblox::Session->new(
                                     master   => $host_ip,
                                     username => "admin",
                                     password => "infoblox"
                                    );
unless($session){
    die("Constructor for session failed: ",
        Infoblox::status_code(). ":" . Infoblox::status_detail());
}
print "Session created successfully.\n";

#Create a DNS zone
my $memberns1 = Infoblox::DNS::Member->new(
                                           name     => "infoblox.localdomain",
                                           ipv4addr => $host_ip,
                                           lead     => "false",
                                           stealth  => "false"
                                          );

#Add the zone
my $firstzone = Infoblox::DNS::Zone->new(
                                         name        => "test.com",
                                         email       => "admin\@infoblox.com",
                                         comment     => "add a zone test.com",
                                         primary     => $memberns1,
                                        );

unless($firstzone){
```

```
    die("Construct test.com zone object failed: ",
        Infoblox::status_code(). ":" .Infoblox::status_detail());
}
print "test.com zone object created successfully.\n";

$session->add($firstzone)
  or die("Add zone for test.com failed: ",
         $session->status_code(). ":" .$session->status_detail());

print"Zone test.com added successfully.\n";

#Export the newly created zone
$session->export_data(
                      type        => 'csv',
                      path        => '/tmp/zone.csv',
                      object_type => 'Infoblox::DNS::Zone',
                      separator   => 'semicolon',

                      # search parameters
                      name        => '.*test.com',
                                        )
                         or die("Export zone for test.com failed: ",
                                $session->status_code(). ":" .$session->status_detail());

# Wait for the export to complete
sleep(5);

print "The zone was exported\n";

# Remove the zone from the database
$session->remove($firstzone)
  or die("Remove zone for test.com failed: ",
         $session->status_code(). ":" .$session->status_detail());

print "The zone was removed\n";

# Import the zone by using the CSV file that was exported

my $import_id = $session->import_data(
                                      type => 'csv',
                                      path => "/tmp/zone.csv",
                                      separator   => 'semicolon',
                                     );

unless($import_id){
    die("Import failed: ",
        Infoblox::status_code(). ":" .Infoblox::status_detail());
}

# Wait for the import to complete
sleep(5);

my $status = $session->get(
                           object  => 'Infoblox::Grid::CSVImportStatus',
                          );

unless($status){
    die("Failed to fetch the import status: ",
        Infoblox::status_code(). ":" .Infoblox::status_detail());
}

print "The zone was re-imported\n";

# The code should be cycling to sleep, fetch and wait until the import is finished.
unless($status->status() eq 'Completed'){
    die("The import took too long to complete: " . $status->status());
}

unless($status->lines_failed() == 0){
    die("There were some import failures: " . $status->lines_failed());
}

print "The zone re-import was successful\n";

# The zone was re-imported. If we import the file again without merging,
# an error occurs.

$import_id = $session->import_data(
                                   type => 'csv',
```

```
                                    path => "/tmp/zone.csv",
                                    separator   => 'semicolon',
                                   );

unless($import_id){
    die("Import failed: ",
        Infoblox::status_code(). ":" .Infoblox::status_detail());
}

# Wait for the import to complete
sleep(5);

print "The zone re-re-import was completed\n";

my $status = $session->get(
                           object  => 'Infoblox::Grid::CSVImportStatus',
                          );

unless($status){
    die("Failed to fetch the import status: ",
        Infoblox::status_code(). ":" .Infoblox::status_detail());
}

# The code should be cycling to sleep, fetch and wait until the import is finished.
unless($status->status() eq 'Completed'){
    die("The import took too long to complete: " . $status->status());
}

if($status->lines_failed() == 0){
    die("There were no import failures: " . $status->lines_failed());
}

print "The zone re-re-import correctly failed\n";

# Review the  failure
$session->export_data(
                      type        => 'csv_error_log',
                      path        => '/tmp/error_log.txt',
                      import_id   => $import_id,
                     )
  or die("Export error log failed: ",
         $session->status_code(). ":" .$session->status_detail());

# The failure is in this file
print "Error log for the failed import\n-------\n";
print `cat /tmp/error_log.txt`;

#Remove the zone
$session->remove(
                 object => "Infoblox::DNS::Zone",
                 name   => "test.com",
                )
  or die("Remove zone test.com failed: ",
         $session->status_code(). ":" .$session->status_detail());

print "The zone was removed\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::DateTime - Grid date and time object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - date( )
  - time( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::DateTime - Grid date and time object.

DESCRIPTION

Grid DateTime object is used to set the date and the time for an Infoblox appliance.

CONSTRUCTOR

```
 #Constructor for an grid DateTime object
 my $date_time = Infoblox::Grid::DateTime->new (
         date    =>  $date            #Optional / Default is undefined and continue to ues current system
date
         time    =>  $time            #Optional / Default is undefined and continue to use current system
time
     );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a grid DateTime object.

Infoblox::Session->get( )

Use this method to retrieve existing objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get a grid DateTime object:

none.
```

Example

```
my @result_array = $session->get( "object" => "Infoblox::Grid::DateTime");
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the date and time of Infoblox appliance.
$object->date( "11/04/2007" );
$object->time( "15:00:00" );
# Submit modification
my $response = $session->modify( $object );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute value of a date time.

date( )

Use this method to set or retrieve date for an Infoblox appliance.

Parameter

Desired date in mm/dd/yy format eg (11/04/2007).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get date
my $date = $object->date();
#Modify ntp_enabled
$object->date( "11/04/2007" );
```

time( )

Use this method to set or retrieve the time for an Infoblox appliance. Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired time in hh/mm/ss format eg. (11:59:59).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get time
my $time = $object->time();
#Modify time
$object->time("15:00:00");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, modify, and remove an object. This sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my @result_array = $session->get( "object" => "Infoblox::Grid::DateTime");
if( defined @result_array  and scalar( @result_array ) > 0 )
 {
        my $object = $result_array[0];
        unless ($object) {
                die("Get on Date Time failed: ",
         $session->status_code() . ":" . $session->status_detail());
        }

$object->date( "11/07/2007" );
$object->time( "15:00:00" );
$session->modify($object)
   or die("Modify on Date Time failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Grid Date Time modified successfully \n";
}
```

```
else
{
        print "No  DateTime Structure found !!!\n";
}

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::DHCP - Grid DHCP services object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - authority( )
  - bootfile( )
  - bootserver( )
  - capture_hostname( )
  - ddns_generate_hostname( )
  - ddns_retry_interval( )
  - ddns_server_always_updates( )
  - ddns_ttl( )
  - ddns_update_fixed_addresses( )
  - ddns_use_option81( )
  - deny_bootp( )
  - email_list( )
  - enable_ddns( )
  - enable_dhcp_thresholds( )
  - enable_email_warnings( )
  - enable_gss_tsig( )
  - enable_ifmap( )
  - enable_roaming_hosts( )
  - enable_snmp_warnings( )
  - grid( )
  - gss_tsig_key( )
  - high_water_mark( )
  - ifmap_port( )
  - ifmap_url( )
  - ignore_dhcp_option_list_request( )
  - ipv6_capture_hostname( )
  - ipv6_ddns_enable_option_fqdn( )
  - ipv6_ddns_server_always_updates( )
  - ipv6_ddns_domainname( )
  - ipv6_ddns_ttl( )
  - ipv6_default_prefix( )
  - ipv6_domain_name( )
  - ipv6_domain_name_servers( )
  - ipv6_enable_ddns( )
  - ipv6_enable_gss_tsig( )
  - ipv6_enable_retry_updates( )
  - ipv6_generate_hostname( )
  - ipv6_gss_tsig_key( )
  - ipv6_kdc_server( )
  - ipv6_microsoft_code_page( )
  - ipv6_options( )
  - ipv6_prefixes( )
  - ipv6_recycle_leases( )
  - ipv6_retry_updates_interval( )
  - ipv6_txt_record_handling( )
  - ipv6_update_dns_on_lease_renewal( )
  - kdc_server( )
  - lease_logging_member( )
  - lease_scavenge_time( )
  - log_lease_events( )
  - low_water_mark( )
  - microsoft_code_page( )
  - nextserver( )
  - option60_match_rules( )
  - options( )
  - ping_count( )
  - ping_timeout( )
  - preferred_lifetime( )
  - pxe_lease_time( )
  - recycle_leases( )
  - retry_ddns_updates( )
  - syslog_facility( )
  - update_dns_on_lease_renewal( )

- valid_lifetime( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::DHCP - Grid DHCP services object.

DESCRIPTION

When you configure an Infoblox appliance as a full-featured DHCP server, you can set operational parameters at the grid, member, network, DHCP range and individual IP address levels. This class supports grid-level DHCP services management and network configuration. It implements methods for configuring grid-level DHCP parameters, such as DHCP general properties, ping and DHCP lease settings, Bootstrap Protocol (BOOTP) properties, custom DHCP options, and DHCP logging.

CONSTRUCTOR

The Infoblox::Grid::DHCP object is automatically generated upon the successful installation of the Infoblox appliance. It does not require manual construction.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a grid DHCP object.

Infoblox::Session->get( )

Use this method to retrieve matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get a specific grid DHCP object:

 grid - Required. A grid name in string format.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DHCP",
    grid   => "Infoblox"
    );
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify authority.
$griddhcp->authority("true");
#Submit modification
my $response = $session->modify( $griddhcp );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of the DHCP properties of a grid.

authority( )

Use this method to set or retrieve the grid-level authority flag. This flag specifies whether a DHCP server is authoritative for a domain.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the authority flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authority
my $authority = $griddhcp->authority();
#Modify authority
$griddhcp->authority("true");
```

bootfile( )

Some DHCP clients use BOOTP (bootstrap protocol) or include the boot file name option in their DHCPREQUEST messages. Use this method to set or retrieve a boot file name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the file that the client must download.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootfile
my $bootfile = $griddhcp->bootfile();
#Modify bootfile
$griddhcp->bootfile("bootfile.txt");
```

bootserver( )

Use this method to set or retrieve the name of the server on which a boot file is stored.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootserver
my $bootserver = $griddhcp->bootserver();
#Modify bootserver
$griddhcp->bootserver("hostname.com");
```

capture_hostname( )

Use this method to set or retrieve the grid-level capture_hostname flag. Set this flag to capture the hostname and lease time when assigning a fixed address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get capture_hostname
my $capture_hostname = $griddhcp->capture_hostname();
```

```
#Modify disable
$griddhcp->capture_hostname("true");
```

ddns_generate_hostname( )

Use this method to set or retrieve the grid-level ddns_generate_hostname flag. Set this flag to enable a DHCP server to generate a host name and update DNS with this host name when it receives a DHCP REQUEST message that does not include a host name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_generate_hostname
my $ddns_generate_hostname = $griddhcp->ddns_generate_hostname();
#Modify disable
$griddhcp->ddns_generate_hostname("true");
```

ddns_retry_interval( )

Use this method to set or retrieve the grid-level ddns_retry_interval value. Set the retry interval when the DHCP server makes repeated attempts to send DDNS updates to a DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the retry interval in minutes. The default retry interval is five minutes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_retry_interval
my $ddns_retry_interval = $griddhcp->ddns_retry_interval();
#Modify ddns_retry_interval
$griddhcp->ddns_retry_interval(10);
```

ddns_server_always_updates( )

Use this method to set or retrieve the grid-level ddns_server_always_updates flag. Set this flag to allow only the DHCP server to update DNS, regardless of requests from DHCP clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ddns_server_always_updates flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_server_always_updates
my $ddns_server_always_updates = $griddhcp->ddns_server_always_updates();
#Modify ddns_server_always_updates
$griddhcp->ddns_server_always_updates("true");
```

ddns_ttl( )

Use this method to set or retrieve the grid-level ddns_ttl value. The DDNS TTL (Dynamic DNS Time To Live) value specifies the number of seconds a DNS record is cached.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that specifies the number of seconds a DNS record is cached. Default value is half of lease time.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_ttl
my $ddns_ttl = $griddhcp->ddns_ttl();
#Modify ddns_ttl
$griddhcp->ddns_ttl(1000);
```

ddns_update_fixed_addresses( )

Use this method to set or retrieve the grid-level ddns_update_fixed_addresses flag. Set this flag to enable the DHCP server to update the A and PTR records of clients with fixed addresses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the DHCP server to update the A and PTR records of clients with fixed addresses, or specify "false" to disable this feature. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_update_fixed_addresses
my $ddns_update_fixed_addresses = $griddhcp->ddns_update_fixed_addresses();
#Modify ddns_update_fixed_addresses
$griddhcp->ddns_update_fixed_addresses("true");
```

ddns_use_option81( )

Use this method to set or retrieve the grid-level ddns_use_option81 flag. Set this flag to enable support for option 81.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable support for option 81 or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_use_option81
my $ddns_use_option81 = $griddhcp->ddns_use_option81();
#Modify ddns_use_option81
$griddhcp->ddns_use_option81("true");
```

deny_bootp( )

Use this method to set or retrieve the grid-level deny_bootp flag. Set this flag to enable a BOOTP server to deny BOOTP request.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $griddhcp->deny_bootp();
#Modify deny_bootp
$griddhcp->deny_bootp("true");
```

email_list( )

Use this method to specify or retrieve the grid-level email_list value. Specify an e-mail address to which you want the Infoblox appliance to send e-mail notifications when the DHCP address usage for the grid crosses a threshold. You can create a list of several e-mail addresses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference with the e-mail values.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get email_list
my $email_list = $griddhcp->email_list();
#Modify email_list
$griddhcp->email_list(["admin\@infoblox.com"]);
```

enable_ddns( )

Use this method to set or retrieve the grid-level enable_ddns flag. Set this flag to enable DHCP servers in a grid to send DDNS updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable support for DDNS updates or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ddns
my $enable_ddns= $griddhcp->enable_ddns();
#Modify enable_ddns
$griddhcp->enable_ddns("true");
```

enable_dhcp_thresholds( )

Use this method to set or retrieve the grid-level enable_dhcp_thresholds flag. Thresholds represent the watermarks above or below which address usage in a network is unexpected and might warrant your attention.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable DHCP thresholds or "false" to disable them. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcp_thresholds
my $enable_dhcp_thresholds = $griddhcp->enable_dhcp_thresholds();
#Modify enable_dhcp_thresholds
$griddhcp->enable_dhcp_thresholds("true");
```

enable_email_warnings( )

Use this method to set or retrieve the grid-level enable_email_warnings flag. If you enable this feature, the Infoblox appliance sends an e-mail notification to an administrator if DHCP address usage crosses a high or low watermark threshold.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_email_warnings flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_email_warnings
my $enable_email_warnings = $griddhcp->enable_email_warnings();
#Modify enable_email_warnings
$griddhcp->enable_email_warnings("true");
```

enable_gss_tsig( )

Use this method to retrieve the grid-level enable_gss_tsig flag. This method returns "true" only if both the gss_tsig_key and kdc_server are specified.

This is for DHCPv4 GSS-TSIG authentication.

Parameter

This is a read-only method. You cannot enter any parameter.

Returns

This method returns the enable_gss_tsig attribute value.

Example

```
#Get enable_gss_tsig
my $enable_gss_tsig = $griddhcp->enable_gss_tsig();
```

enable_ifmap( )

Use this method to enable or disable the IF-MAP DHCP client feature on this grid.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Boolean to enable or disable IF-MAP client feature. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get IF-MAP enable value
my $ifmap_enabled = $griddhcp->enable_ifmap();
#Enable IF-MAP Client feature
$griddhcp->enable_ifmap("true");
```

enable_roaming_hosts( )

Use this method to set or retrieve the grid-level enable_roaming_hosts flag. Set this flag to enable DHCP servers in a grid to support roaming hosts.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_roaming_hosts flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_roaming_hosts
my $enable_roaming_hosts = $griddhcp->enable_roamin_hosts();
#Modify enable_roaming_hosts
$griddhcp->enable_roaming_hosts("true");
```

enable_snmp_warnings( )

Use this method to set or retrieve the grid-level enable_snmp_warnings flag. When DHCP address usage crosses a watermark threshold, the appliance sends an SNMP trap to trap receivers that you defined for the grid.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_snmp_warnings flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_snmp_warnings
my $enable_snmp_warnings = $griddhcp->enable_snmp_warnings();
#Modify enable_snmp_warnings
$griddhcp->enable_snmp_warnings("true");
```

grid( )

Use this method to retrieve the grid that serves DHCP. This specifies a group of Infoblox appliances that are connected together to provide a single point of device administration and service configuration in a secure, highly available environment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name of the grid that provides DHCP service. Default value is "Infoblox".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#get grid
my $grid = $griddhcp->grid();
```

gss_tsig_key( )

Use this method to set or retrieve the DHCPv4 GSS-TSIG key.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Infoblox::Grid::KerberosKey object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get gss_tsig_key
my $gss_tsig_key = $griddhcp->gss_tsig_key();

#set gss_tsig_key
my $kerberos = Infoblox::Cluster::KerberosKey->new(
       principal => 'host/appliance.ivan.test@LADYBUG.LOCAL',
       );
$griddhcp->gss_tsig_key($kerberos);
$session->modify($griddhcp);
```

high_water_mark( )

Use this method to set or retrieve the grid-level high_water_mark value. If the percent of allocated addresses exceeds this watermark, the appliance makes a syslog entry and if enabled, sends an e-mail notification.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number with the percentage value. Range is from 1 to 100. The high watermark value must be greater than the low watermark value. Default value is 95.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get high_water_mark
my $high_water_mark = $griddhcp->high_water_mark();
#Modify high_water_mark
$griddhcp->high_water_mark(80);
```

ifmap_port( )

Use this method to set or retrieve the IF-MAP server port used by DHCP IF-MAP clients. This setting is optional and defaults to the port associated with the protocol specified in ifmap_url (eg: for HTTP, the port is 80; for HTTPS, the port is 443).

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The IF-MAP server port number. The default value is undef.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get IF-MAP server port
my $port = $griddhcp->ifmap_port();
#Modify IF-MAP server port
$griddhcp->ifmap_port(1234);
```

ifmap_url( )

Use this method to set or retrieve the IF-MAP server URL used by DHCP IF-MAP clients. This setting is mandatory if IF-MAP is enabled on this grid.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

String for the IF-MAP server URL. The default value is undef.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get IF-MAP server URL
my $url = $griddhcp->ifmap_url();
#Modify IF-MAP server URL
$griddhcp->ifmap_url("https://ifmap.infoblox.com");
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the grid-level ignore_dhcp_option_list_request flag. If this flag is set to true all available DHCP options will be returned to the client.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $griddhcp->ignore_dhcp_option_list_request();
#Modify ignore_dhcp_option_list_request
$griddhcp->ignore_dhcp_option_list_request("true");
```

ipv6_capture_hostname( )

Use this method to set or retrieve the grid-level ipv6_capture_hostname flag. Set this flag to capture the ipv6 hostname and lease time when assigning a fixed address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_capture_hostname
my $ipv6_capture_hostname = $griddhcp->ipv6_capture_hostname();
#Modify disable
$griddhcp->ipv6_capture_hostname("true");
```

ipv6_ddns_enable_option_fqdn( )

Use this method to set or retrieve the grid-level ipv6_ddns_enable_option_fqdn flag. This method controls whether the FQDN option sent by the client is to be used, or if the server can automatically generate the FQDN.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_enable_option_fqdn
my $ipv6_ddns_enable_option_fqdn = $griddhcp->ipv6_ddns_enable_option_fqdn();
#Modify disable
```

```
$griddhcp->ipv6_ddns_enable_option_fqdn("true");
```

ipv6_ddns_server_always_updates( )

Use this method to set or retrieve the grid-level ipv6_ddns_server_always_updates flag. Set this flag to "false" to update DNS only if requested by the client.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_server_always_updates
my $ipv6_ddns_server_always_updates = $griddhcp->ipv6_ddns_server_always_updates();
#Modify ipv6_ddns_server_always_updates
$griddhcp->ipv6_ddns_server_always_updates("true");
```

ipv6_ddns_domainname( )

Use this method to set or retrieve the grid-level IPv6 DDNS domain name value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DDNS domain name in FQDN (Fully Qualified Domain Name) format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_domainname
my $ipv6_ddns_domainname = $griddhcp->ipv6_ddns_domainname();
#Modify ipv6_ddns_domainname
$griddhcp->ipv6_ddns_domainname("domain.com");
```

ipv6_ddns_ttl( )

Use this method to set or retrieve the grid-level IPv6 DDNS TTL.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The preferred DDNS TTL value in seconds. The default is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_ttl
my $ipv6_ddns_ttl = $griddhcp->ipv6_ddns_ttl();
#Modify ipv6_ddns_ttl
$griddhcp->ipv6_ddns_ttl(12345);
```

ipv6_default_prefix( )

Use this method to set or retrieve the grid-level IPv6 default prefix.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The desired IPv6 prefix. It must be one of those listed in the ipv6_prefixes method.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_default_prefix
my $ipv6_default_prefix = $griddhcp->ipv6_default_prefix();
#Modify ipv6_default_prefix
$griddhcp->ipv6_default_prefix("2001::");
```

ipv6_domain_name( )

Use this method to set or retrieve the ipv6_domain_name value for a Grid DHCP object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the client's domain for resolving names in DNS in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_domain_name
my $ipv6_domain_name = $griddhcp->ipv6_domain_name();
#Modify ipv6_domain_name
$griddhcp->ipv6_domain_name("test_domain.com");
```

ipv6_domain_name_servers( )

Use this method to set or retrieve the ipv6_domain_name_servers value of a Grid DHCP object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The comma-separated domain name server addresses in IPv6 address format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_domain_name_servers
my $ipv6_domain_name_servers = $griddhcp->ipv6_domain_name_servers();
#Modify ipv6_domain_name_servers
$griddhcp->ipv6_domain_name_servers("2001::1,2001::2");
```

ipv6_enable_ddns( )

Use this method to set or retrieve the grid-level ipv6_enable_ddns flag. Set this flag to enable DHCPv6 dynamic DNS updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_enable_ddns
my $ipv6_enable_ddns = $griddhcp->ipv6_enable_ddns();
#Modify ipv6_enable_ddns
$griddhcp->ipv6_enable_ddns("true");
```

ipv6_enable_gss_tsig( )

Use this method to retrieve the grid-level ipv6_enable_gss_tsig flag. This method returns "true" only if both the ipv6_gss_tsig_key and ipv6_kdc_server are specified.

This is for DHCPv6 GSS-TSIG authentication.

Parameter

This is a read-only method. You cannot enter any parameter.

Returns

This method returns the ipv6_enable_gss_tsig attribute value.

Example

```
#Get ipv6_enable_gss_tsig
my $ipv6_enable_gss_tsig = $griddhcp->ipv6_enable_gss_tsig();
```

ipv6_enable_retry_updates( )

Use this method to set or retrieve the ipv6_enable_retry_updates flag. If this flag is set to "true", the DHCPv6 server retries failed dynamic DNS updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the DHCP server to retry failed DNS updates or "false" to disable it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_enable_retry_updates
my $ipv6_enable_retry_updates = $griddhcp->ipv6_enable_retry_updates();
#Modify ipv6_enable_retry_updates
$griddhcp->ipv6_enable_retry_updates("true");
```

ipv6_generate_hostname( )

Use this method to set or retrieve the grid-level ipv6_generate_hostname flag. Set this flag to generate the hostname if it is not sent by the client.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_generate_hostname
my $ipv6_generate_hostname = $griddhcp->ipv6_generate_hostname();
#Modify ipv6_generate_hostname
$griddhcp->ipv6_generate_hostname("true");
```

ipv6_gss_tsig_key( )

Use this method to set or retrieve the DHCPv6 GSS-TSIG key.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Infoblox::Grid::KerberosKey object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_gss_tsig_key
my $ipv6_gss_tsig_key = $griddhcp->ipv6_gss_tsig_key();

#Set ipv6_gss_tsig_key
my $kerberos = Infoblox::Cluster::KerberosKey->new(
       principal => 'host/appliance.ivan.test@LADYBUG.LOCAL',
       );
$griddhcp->ipv6_gss_tsig_key($kerberos);
$session->modify($griddhcp);
```

ipv6_kdc_server( )

Use this method to specify or retrieve the IPv4 address or FQDN of the Kerberos server for DCHPv6 GSS-TSIG authentication.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The Kerberos server IPv4 address or FQDN, with a maximum of 1023 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_kdc_server
my $ipv6_kdc_server = $griddhcp->ipv6_kdc_server();
#Modify ipv6_kdc_server
$griddhcp->ipv6_kdc_server("10.32.0.8");
```

ipv6_microsoft_code_page( )

Use this method to set or retrieve the grid-level Microsoft client DHCP IPv6 code page value. This value is the hostname translation code page for Microsoft DHCP IPv6 clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Microsoft DHCP client code page in string format. Valid values are:

None

Arabic (1256)

Arabic (ISO-8859-6)

Baltic (1257)

Baltic (775)

Baltic (ISO-8859-4)

Central Europe (1250)

Cyrillic (1251)

Cyrillic (855)

Cyrillic (ISO-8859-5)

Greek (1253)

Greek (737)

Greek (ISO-8859-7)

Hebrew (1255)

Hebrew (862)

Hebrew (ISO-8859-8)

Japanese Shift-JIS (932)

Korean (949)

Latin 1 (ISO-8859-1)

Latin 2 (ISO-8859-2)

Latin 3 (ISO-8859-3)

Latin 9 (ISO-8859-15)

Latin I (1252)

Latin II (852)

Multilingual Latin I (850)

Russian (866)

Simplified Chinese GBK (936)

Thai (874)

Traditional Chinese Big5 (950)

Turkish (1254)

Turkish (857)

Turkish (ISO-8859-9)

US (437)

Vietnam (1258)

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_microsoft_code_page
my $ipv6_microsoft_code_page = $griddhcp->ipv6_microsoft_code_page();
#Modify ipv6_microsoft_code_page
$griddhcp->ipv6_microsoft_code_page("Cyrillic (ISO-8859-5)");
```

ipv6_options( )

Use this method to set or retrieve the grid-level IPv6 options.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_options
my $ipv6_options = $griddhcp->ipv6_options();
#Modify ipv6_options
```

```
my $option1 = Infoblox::DHCP::Option->new(
    name   => "dhcp6.rapid-commit",
    value  => "true"
);
$griddhcp->ipv6_options([$option1]);
```

ipv6_prefixes( )

Use this method to set or retrieve the grid-level list of IPv6 prefixes.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A comma-separated list of IPv6 prefixes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_prefixes
my $ipv6_prefixes = $griddhcp->ipv6_prefixes();
#Modify ipv6_prefixes
$griddhcp->ipv6_prefixes("2001::,2002::");
```

ipv6_recycle_leases( )

Use this method to set or retrieve the ipv6_recycle_leases flag of a grid DHCP server. If you enable this feature, and then delete an IPv6 DHCP range, the appliance stores active leases from this range up to one week after the leases expire.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ipv6_recycle_leases flag or "false" to deactivate/unset it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_recycle_leases
my $ipv6_recycle_leases = $griddhcp->ipv6_recycle_leases();
#Modify ipv6_recycle_leases
$griddhcp->ipv6_recycle_leases("true");
```

ipv6_retry_updates_interval( )

Use this method to set or retrieve the ipv6_retry_updates_interval value. Set the retry interval when the member DHCPv6 server makes repeated attempts to send DDNS updates to a DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the retry interval in minutes. The default retry interval is five minutes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_retry_updates_interval
my $ipv6_retry_updates_interval = $griddhcp->ipv6_retry_updates_interval();
#Modify ipv6_retry_updates_interval
$griddhcp->ipv6_retry_updates_interval(10);
```

ipv6_txt_record_handling( )

Use this method to set or retrieve the grid-level TXT record handling value. This value specifies how DHCP should treat TXT records when performing DNS update

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

TXT record handling value in string format. Valid values are: "IGNORE_CONTENTS", "ISC", "ISC_TRANSITIONAL" and "MS". The default value is "ISC".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_txt_record_handling
my $ipv6_txt_record_handling = $griddhcp->ipv6_txt_record_handling();
#Modify ipv6_txt_record_handling
$griddhcp->ipv6_txt_record_handling("ISC_TRANSITIONAL");
```

ipv6_update_dns_on_lease_renewal( )

The ipv6_update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when an IPv6 DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ipv6_update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_update_dns_on_lease_renewal
my $ipv6_update_dns_on_lease_renewal=$griddhcp->ipv6_update_dns_on_lease_renewal( );
#Modify ipv6_update_dns_on_lease_renewal
$griddhcp->ipv6_update_dns_on_lease_renewal("true");
```

kdc_server( )

Use this method to specify or retrieve the IPv4 address or FQDN of the Kerberos server for DCHPv4 GSS-TSIG authentication.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The Kerberos server IPv4 address or FQDN, with a maximum of 1023 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get kdc_server
my $kdc_server = $griddhcp->kdc_server();
#Modify  kdc_server
$griddhcp->kdc_server("10.32.0.7");
```

lease_logging_member( )

Use this method to specify or retrieve the grid member on which you want to store the DHCP lease history log. Infoblox recommends that you dedicate a member other than the master as a logging member. If possible, use this member solely for storing the DHCP lease history log. If you do not select a member, no logging can occur.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Member object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lease_logging_member
my $lease_logging_member = $griddhcp->lease_logging_member();
#Modify lease_logging_member
my $dhcpmember = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2"
);

$griddhcp->lease_logging_member($dhcpmember);
```

lease_scavenge_time( )

Use this method to set or retrieve the grid-level lease_scavenge_time value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An integer that specifies the period of time, in seconds, that free and backup leases remain in the database before they are automatically deleted. To disable lease scavenging, set the parameter to -1. The minimum positive value must be greater than 86400 seconds (1 day). The default value of this parameter is -1, which means that lease scavenging is disabled.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lease_scavenge_time
my $lease_scavenge = $griddhcp->lease_scavenge_time();
#Modify lease_scavenge_time
$griddhcp->lease_scavenge_time(30 * 24 * 60 * 60);
```

log_lease_events( )

Use this method to set or retrieve the grid-level log_lease_events flag. This value specifies whether grid members log lease events.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the log_lease_events flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get log_lease_events
my $log_lease_events = $griddhcp->log_lease_events();
#Modify log_lease_events
$griddhcp->log_lease_events("true");
```

low_water_mark( )

Use this method to set or retrieve the grid-level low_water_mark value. If the percent of allocated addresses drops below this watermark, the appliance makes a syslog entry and if enabled, sends an e-mail notification.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number with the percentage value. Range is from 0 to 100. The low watermark value must be lower than the high watermark value. The default low watermark number is 0.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get low_water_mark
my $low_water_mark = $griddhcp->low_water_mark();
#Modify low_water_mark
$griddhcp->low_water_mark(10);
```

microsoft_code_page( )

Use this method to set or retrieve the grid-level Microsoft client DHCP IPv4 code page value. This value is the hostname translation code page for Microsoft DHCP IPv4 clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Microsoft DHCP client code page in string format. Valid values are the same as those for ipv6_microsoft_code_page. See ipv6_microsoft_code_page().

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get microsoft_code_page
my $microsoft_code_page = $griddhcp->microsoft_code_page();
#Modify microsoft_code_page
$griddhcp->microsoft_code_page("Cyrillic (ISO-8859-5)");
```

nextserver( )

Use this method to set or retrieve the grid-level "next server" value. This value is the IP address or name of the boot file server on which the boot file is stored.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The boot file server IP address or name in FQDN format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot file server name can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nextserver
my $nextserver = $griddhcp->nextserver();
#Modify nextserver
$griddhcp->nextserver("1.1.1.1");
```

option60_match_rules( )

Use this method to set or retrieve grid-level DHCP option60 match rules. An Option60MatchRule object allows specifying match rules for Option 60 globally.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option60MatchRule object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get option60_match_rules
my $option_matchrule = $griddhcp->option60_match_rules();
#Modify option60_match_rules
my $option60_match_rule = Infoblox::DHCP::Option60MatchRule->new(
                                    "option_space" => "infospace",
                                    "match_value" => "Nortel-i2004-A",
                                    "is_substring" => "true",
                                    "substring_offset" => "8",
                                    "substring_length" => "14"
                                  );

# Configure DHCP Option60MatchRule on the Grid DHCP object
$griddhcp->option60_match_rules([$option60_match_rule]);

#Remove Option60MatchRule object
$griddhcp->option60_match_rules([]);
```

options( )

Use this method to set or retrieve grid-level DHCP options. These options provide network configuration settings and various services available on the network.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $griddhcp->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name   => "lease-time",
    value  => 50
);

my $option2 = Infoblox::DHCP::Option->new(
    name    => "routers",
    value   => "1.4.1.1,2.2.2.2",
);

my $option3 = Infoblox::DHCP::Option->new(
    name     => "nameservers",
    value    => "1.1.1.1, 2.2.2.2",
);

my $option4 = Infoblox::DHCP::Option->new(
    name  => "domain-name",
    value => "domainname.com",
);

my $option5 = Infoblox::DHCP::Option->new(
    name     => "broadcast",
    value    => "255.0.0.0",
);

$griddhcp->options([$option1, $option2, $option3, $option4, $option5]);
```

ping_count( )

Use this method to set or retrieve the grid-level ping_count value. This value specifies the number of pings that the Infoblox appliance

sends to an IP address to verify that it is not in use.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number of pings. Range is from 0 to 10, where 0 disables pings. The default value is 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ping_count
my $ping_count = $griddhcp->ping_count();
#Modify ping_count
$griddhcp->ping_count(4);
```

ping_timeout( )

Use this method to set or retrieve the grid-level ping_timeout value. This value indicates the number of seconds the appliance waits for a response to its ping.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Ping timeout value in seconds. Range is from 1 to 5 seconds. The default value is 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ping_timeout
my $ping_timeout = $griddhcp->ping_timeout();
#Modify ping_timeout
$griddhcp->ping_timeout(3);
```

preferred_lifetime( )

Use this method to set or retrieve the grid-level preferred lifetime.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Preferred lifetime value in seconds. The default value is 27000.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get preferred_lifetime
my $preferred_lifetime = $griddhcp->preferred_lifetime();
#Modify preferred_lifetime
$griddhcp->preferred_lifetime(12345);
```

pxe_lease_time( )

Use this method to set or retrieve the grid-level pxe_lease_time value. This value is the duration of time it takes a host to connect to a boot server, such as a TFTP server, and download the file it needs to boot.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that specifies the lease time in seconds. Default value is 43200 seconds.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get pxe_lease_time
my $pxe_lease_time = $griddhcp->pxe_lease_time();
#Modify pxe_lease_time
$griddhcp->pxe_lease_time(200);
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases flag of a grid DHCP server. If you enable this feature, and then delete a DHCP range, the appliance stores active leases from this range up to one week after the leases expire.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the recycle_leases flag or "false" to deactivate/unset it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $griddhcp->recycle_leases();
#Modify recycle_leases
$griddhcp->recycle_leases("true");
```

retry_ddns_updates( )

Use this method to set or retrieve the grid-level retry_ddns_updates flag. This value indicates whether the grid DHCP servers make repeated attempts to send DDNS updates to a DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the DHCP servers to retry DDNS updates or "false" to disable this feature. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get retry_ddns_updates
my $retry_ddns_updates = $griddhcp->retry_ddns_updates();
#Modify retry_ddns_updates
$griddhcp->retry_ddns_updates("true");
```

syslog_facility( )

Use this method to set or retrieve the grid-level syslog_facility value. The syslog facility is the location on the syslog server to which you want to sort syslog messages.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are: `"daemon"`, `"local0"`, `"local1"`, `"local2"`, `"local3"`, `"local4"`, `"local5"`, `"local6"`, `"local7"`. The default value is "daemon".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get syslog_facility
my $syslog_facility = $griddhcp->syslog_facility();
#Modify syslog_facility
$griddhcp->syslog_facility("local0");
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$griddhcp->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$griddhcp->update_dns_on_lease_renewal("true");
```

valid_lifetime( )

Use this method to set or retrieve the valid lifetime value on the Grid level.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid lifetime value in seconds. The default value is 43200.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get valid_lifetime
my $valid_lifetime = $roaming_host->valid_lifetime();
#Modify valid_lifetime
$roaming_host->valid_lifetime(12345);
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to a grid DHCP object, such as get and modify an object. This sample code also includes error handling for the operations.

#Preparation prior to get and modify operations on a grid DHCP object

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#refers to Infoblox Appliance IP address
my $host_ip   = "192.168.1.2";
my $host_name = "infoblox.localdomain";

#Creating a session to appliance.
my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
    )
    or die("Constructor for session failed: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());
```

```
print"Session created successfully.\n";

#Creating a domain-name option object.
my $option = Infoblox::DHCP::Option->new(
                name   => "domain-name",
                value  => "domainname.com",
)
or die("Constructor for option failed: ",
              Infoblox::status_code(). ":" .Infoblox::status_detail());

print"dhcp option object created successfully.\n";

#Creating a Member object.
my $dhcpmember = Infoblox::DHCP::Member->new(
    name     => $host_name,
    ipv4addr => $host_ip
    )
    or die("Constructor for dhcpmember failed: ",
              Infoblox::status_code(). ":" .Infoblox::status_detail());

print"dhcp member object created successfully.\n";

#Creating a forward zone nameserver.
my $ns1 = Infoblox::DNS::Nameserver->new(
    ddns_zone => "sub.test.com",
    name      => "server.dns.com",
    ipv4addr  => "1.1.1.1",
    TSIGname  => "tsig_name",
    TSIGkey   => "ZX1jVJl7C58GT/sc7Q3ucA=="
    )
    or die("Constructor for ns1 failed: ",
              Infoblox::status_code(). ":" .Infoblox::status_detail());

print"forward zone nameserver object created successfully.\n";

#Creating a reverse zone nameserver.
my $ns2 = Infoblox::DNS::Nameserver->new(
    ddns_zone => "10.0.0.0/8",
    name      => "server.dns.com",
    ipv4addr  => "1.1.1.1"
    )
    or die("Constructor for ns2 failed: ",
              Infoblox::status_code(). ":" .Infoblox::status_detail());

print"reverse zone nameserver object created successfully.\n";
```

#Get and modify the grid DHCP object.

```
#Getting the DHCP object from appliance through session.
my @result = $session->get(
    object => "Infoblox::Grid::DHCP",
    grid   => "Infoblox"
    )
    or die("get DHCP failed: ",
              $session->status_code(). ":" .$session->status_detail());
print "get grid DHCP Objcet successful.\n";

unless (scalar(@result) == 0) {
    my $griddhcp = $result[0];
    if ($griddhcp) {

        #Modifying the value of the specified objects.
        $griddhcp->authority("false");
        $griddhcp->enable_ddns("true");
        $griddhcp->bootserver("boot.test.com");
        $griddhcp->ddns_ttl(21600);
        $griddhcp->nextserver("77.77.77.7");
        $griddhcp->pxe_lease_time(43200);
        $griddhcp->bootfile("bootfile");
        $griddhcp->ping_count(3);
        $griddhcp->ping_timeout(2);
        $griddhcp->email_list(["admin\@infoblox.com", "bkp_admin\@infoblox.com"]);
        $griddhcp->options([$option]);
        $griddhcp->lease_logging_member($dhcpmember);
        $griddhcp->microsoft_code_page("Cyrillic (ISO-8859-5)");
```

```
        #Applying the changes to appliance through session.
        $session->modify($griddhcp)
            or die("Modify DHCP failed: ",
               $session->status_code(). ":" .$session->status_detail());
                print"Modified Grid DHCP Object successfully.\n";
              }
         } else {
             print "No Grid DHCP object found.";
         }

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DHCP::Member, Infoblox::DHCP::Option, Infoblox::DNS::Nameserver, Infoblox::Grid::KerberosKey, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::DNS - DNS objects management and configuration

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - allow_bulkhost_ddns( )
  - allow_gss_tsig_zone_updates( )
  - allow_query( )
  - allow_recursive_query( )
  - allow_transfer( )
  - allow_update( )
  - bind_check_names_policy( )
  - blackhole_list( )
  - blacklist_action( )
  - blacklist_log_query( )
  - blacklist_redirect_addresses( )
  - blacklist_redirect_ttl( )
  - blacklist_rulesets( )
  - bulk_host_name_template( )
  - bulk_host_name_templates( )
  - default_ttl( )
  - dns64_groups( )
  - dnssec_enabled( )
  - dnssec_expired_signatures_enabled( )
  - dnssec_ksk_algorithm( )
  - dnssec_ksk_rollover_interval( )
  - dnssec_ksk_size( )
  - dnssec_signature_expiration( )
  - dnssec_validation_enabled( )
  - dnssec_trusted_keys( )
  - dnssec_zsk_algorithm( )
  - dnssec_zsk_rollover_interval( )
  - dnssec_zsk_size( )
  - email( )
  - enable_blackhole( )
  - enable_blacklist( )
  - enable_dns64( )
  - enable_host_rrset_order( )
  - expire_after( )
  - forward_only( )
  - forward_updates( )
  - forwarders( )
  - host_name_restriction_policy( )
  - logging_categories( )
  - logging_facility( )
  - member_secondary_notify( )
  - negative_ttl( )
  - notify_delay( )
  - nxdomain_redirect( )
  - nxdomain_redirect_addresses( )
  - nxdomain_redirect_ttl( )
  - nxdomain_log_query( )
  - nxdomain_rulesets( )
  - notify_source_port( )
  - nsgroup_default( )
  - nsgroups( )
  - preserve_host_rrset_order_on_secondaries( )
  - query_source_port( )
  - record_name_policies( )
  - recursive_query_list( )
  - refresh_timer( )
  - retry_timer( )
  - rootNS( )
  - sortlist( )
  - transfer_excluded_servers( )
  - transfer_format( )
  - zone_deletion_double_confirm( )
- SAMPLE CODE
- AUTHOR

SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::DNS - DNS objects management and configuration

DESCRIPTION

This class supports DNS service management and configuration such as time-to-live (TTL) settings, zone transfers, queries, root name servers, dynamic updates, sort lists, Transaction Signatures (TSIG) for DNS and others, all at the grid level.

The service configurations of a grid are inherited by all members, zones, and networks unless you specifically override them for selected members, zones, and networks. For this reason, it is recommended that you configure services at the grid level before configuring member, zone and network services.

CONSTRUCTOR

The Infoblox::Grid::DNS object is automatically generated upon the successful installation of the Infoblox appliance. It does not require manual construction.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a grid DNS object.

Infoblox::Session->get( )

Use this method to retrieve the matching grid DNS object from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attribute to get a specific grid DNS object:

 name - Required. A grid name in string format.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS",
    name   => "Infoblox"
    );
```

Infoblox::Session->modify( )

Use this method to modify a grid DNS object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use this method to modify the grid  DNS service.
$grid_dns->allow_recursive_query("true");
#Submit modification
my $response = $session->modify( $grid_dns );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a grid DNS object.

allow_bulkhost_ddns( )

Use this method to set or retrieve the allow_bulkhost_ddns flag of a grid DNS object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the allow_bulkhost_ddns flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get  the allow_bulkhost_ddns flag
my $allow_bulkhost_ddns = $grid_dns->allow_bulkhost_ddns();
#Modify allow bulkhost ddns flag
$grid_dns->allow_bulkhost_ddns("true");
```

allow_gss_tsig_zone_updates( )

Use this method to set or retrieve the allow_gss_tsig_zone_updates flag of a grid.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get  the allow_gss_tsig_zone_updates flag
my $allow_gss_tsig_zone_updates = $grid_dns->allow_gss_tsig_zone_updates();
#Modify allow_gss_tsig_zone_updates flag
$grid_dns->allow_gss_tsig_zone_updates("true");
```

allow_query( )

Use this method to set or retrieve the list of allowed/denied queries for a Grid DNS object. The list specifies IP addresses and/or networks from which queries are allowed or denied. The appliance can also use TSIG keys to authenticate the queries.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 or IPv6 addresses, networks and/or Infoblox::DNS::TSIGKey objects. To deny queries, add "!" as a prefix to addresses and networks. Specify "any" to allow queries from any address. By default, queries are allowed from any address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get allow query list
my $allow_query_list = $grid_dns->allow_query();
#Modify allow query list
$grid_dns->allow_query(["any"]);

#Specifying TSIG keys
my $tsig_key1 = $session->gen_tsig_key();
my $query_key = Infoblox::DNS::TSIGKey->new(
    name => "query_key1",
    key  => $tsig_key1
);
#Modify allow query list
$grid_dns->allow_query(["10.0.0.10" , "20.0.0.0/24" ,"2001::123a","2003::222/64", "!"."30.0.0.30",
$query_key, "!"."40.10.0.0/24","!"."2004::125b","!"."2006::34bc"]);
```

allow_recursive_query( )

Use this method to set or retrieve the allow_recursive_query flag of a grid DNS object. If enabled and the Infoblox appliance receives a query for DNS data, it sends a recursive query to the configured internal root servers. If there are no internal root servers configured, the appliance then sends a recursive query to the Internet root servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the allow_recursive_query flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get allow recursive query flag
my $allow_recursive_query = $grid_dns->allow_recursive_query();
#Modify allow recursive query flag
$grid_dns->allow_recursive_query("true");
```

allow_transfer( )

Use this method to set or retrieve the list of allowed/denied zone transfers for a Grid DNS object. The list specifies IP addresses and/or networks from which zone transfers are allowed or denied. It is also possible to specify the TSIG keys of clients that are allowed to perform zone transfers. TSIG keys are used to authenticate zone transfer requests and replies. The same key name and key value must be on the primary and secondary name servers for TSIG-authenticated zone transfers to occur.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IP addresses, networks, and/or Infoblox::DNS::TSIGKey objects. To deny zone transfer, add "!" as a prefix to IP addresses and/or networks. Specify "any" to allow transfers from any IP address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get allow transfer list
my $allow_transfer_list = $grid_dns->allow_transfer();
#Modify allow transfer list
$grid_dns->allow_transfer(["any"]);

#Specifying TSIG keys
my $tsig_key1 = $session->gen_tsig_key();
my $key1 = Infoblox::DNS::TSIGKey->new(
    name => "key1",
    key  => $tsig_key1
);
my $key2 = Infoblox::DNS::TSIGKey->new(
    name => "tsig_xfer",
    key  => ":2xCOMPAT"
    );
#Modify allow transfer list
$grid_dns->allow_transfer(["10.0.0.10", "20.0.0.0/24","2004::325f","2007::876c/64","!"."30.0.0.30",
$key1, $key2, "!"."40.10.0.0/24"]);
```

allow_update( )

Use this method to set or retrieve the list of allowed/denied dynamic updates for a grid DNS object. The list specifies from which IP addresses, networks or TSIG key authenticated hosts dynamic updates are allowed or denied.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IP addresses, networks, and/or Infoblox::DNS::TSIGKey objects. To deny updates, add "!" as a prefix to IP addresses and/or networks. Specify "any" to allow updates from any IP address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get allow update list
my $allow_update_list = $grid_dns->allow_update();
#Modify allow update list
$grid_dns->allow_update(["any"]);

#Creating a TSIG key object.
my $update_key1 = $session->gen_tsig_key();

my $up_key1 = Infoblox::DNS::TSIGKey->new(
    name => "update_key",
```

```
    key  => $update_key1
    );

 #Modify allow update list
 $grid_dns->allow_update(["10.0.0.10", "20.0.0.0/24","2008::129b","!"."30.0.0.30", $up_key1,
"!"."40.10.0.0/24","!"."2004::76bc/64"]);
```

bind_check_names_policy( )

Use this method to set or retrieve the BIND check names policy, which indicates the action the appliance takes when it encounters host names that do not comply with the Strict Hostname Checking policy. This method applies only if the host name restriction policy is set to "Strict Hostname Checking".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The allowed values of the parameter are "fail", "warn" or undefined. This indicates the desired BIND behavior in the case of illegal hostnames. The default value is undefined, which means the appliance ignores illegal hostnames.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
 #Get bind_check_names_policy value
 my $names_policy = $grid_dns->bind_check_names_policy();
 #Modify bind_check_names_policy
 $grid_dns->bind_check_names_policy("warn");
```

blackhole_list( )

Use this method to set or retrieve the blackhole list of a grid DNS object. This list specifies the IP addresses and networks from which the DNS queries are blocked.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A valid value is an array reference that contains IP addresses and networks. To allow queries, add "!" as the prefix to the IP addresses and networks. Specify "any" to block DNS queries from any IP address.

Returns

When you specify a parameter and the modification succeeds, the method returns "true". When the operation fails, the method returns "false".

When you do not specify a parameter, the method returns the attribute value.

Example

```
 #Get the blackhole list
 my $blackhole_list = $grid_dns->blackhole_list();
 #Modify the blackhole_list
 $grid_dns->blackhole_list(["any"]);
 #Modify the blackhole list
 $grid_dns->blackhole_list(["10.0.0.10",
"20.0.0.0/24","2010::256/64","!"."2011::23cb/64","!"."30.0.0.30", "!"."40.10.0.0/24"]);
 #Modify the blackhole list
 $grid_dns->blackhole_list(undef);
```

blacklist_action( )

Use this method to specify the action to perform when a domain name matches the pattern defined in a rule that is specified by the blacklist_ruleset method.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "REDIRECT" or "REFUSE". The default value is "REFUSE".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Examples

```
#Get blacklist_action
my $blacklist_action = $grid_dns->blacklist_action();
#Modify blacklist_action
$grid_dns->blacklist_action("REDIRECT");
```

blacklist_log_query( )

Use this method to set or retrieve the flag that indicates whether blacklist redirection queries are logged.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable logging, or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_log_query
my $blacklist_log_query = $grid_dns->nxdomain_log_query( );
#Modify blacklist_log_query
$grid_dns->blacklist_log_query("true");
```

blacklist_redirect_addresses( )

Use this method to specify the IP addresses the appliance includes in the response it sends in place of a blacklisted IP address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array of IPv4 addresses. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_redirect_addresses
my $blacklist_redirect_addresses = $grid_dns->blacklist_redirect_addresses( );
#Modify blacklist_redirect_addresses
$grid_dns->blacklist_redirect_addresses(["10.0.0.10", "10.0.0.20"]);
```

blacklist_redirect_ttl( )

Use this method to set or retrieve the TTL value of the synthetic DNS responses that result from blacklist redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer, from 0 to 4294967295, that represents the TTL in seconds. The default value is 60.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_redirect_ttl
my $blacklist_redirect_ttl = $grid_dns->blacklist_redirect_ttl( );
#Modify blacklist_redirect_ttl
$grid_dns->blacklist_redirect_ttl(60);
```

blacklist_rulesets( )

Use this method to specify or retrieve Infoblox::DNS::Ruleset object names assigned at the grid level for blacklist redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains the names of the Infoblox::DNS::Ruleset objects assigned to the grid. The names must be in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_rulesets
my $blacklist_rulesets = $grid_dns->blacklist_rulesets();
#Modify blacklist_rulesets
$grid_dns->blacklist_rulesets(["ruleset1", "ruleset2"]);
```

bulk_host_name_template( )

Use this method to set or retrieve the bulk host name format of a grid DNS object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of a valid bulk host name format.

For instance, if the prefix of a bulk host is *info*, the IP address range is *213.19.32.133-213.19.32.134*, and the domain name is *infoblox.com*. If the bulk host name template "Three Octets" is set at the grid level, then the bulk host names will be *info-213-19-23-133.infoblox.com* and *info-213-19-23-134.infoblox.com*.

```
If the bulk host name template specified is "CustomerTemplate"(*#3*#4), the bulk host names will be
I<info*023*133.infoblox.com> and I<info-023-134.infoblox.com>.
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the bulk host name format
my $grid_template = $grid_dns->bulk_host_name_template();
#Modify bulk host name format
$grid_dns->bulk_host_name_template("Three Octets");
```

bulk_host_name_templates( )

Use this method to set or retrieve the list of bulk host name templates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::DNS::BulkHostNameTemplate object(s).

There are four Infoblox predefined bulk host name templates.

Template Name Template Format

"Four Octets" -$1-$2-$3-$4

"Three Octets" -$2-$3-$4

"Two Octets" -$3-$4

"One Octet" -$4

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bulk_host_name_templates
my $bulk_host_name_templates = $grid_dns->bulk_host_name_templates();

#Modify bulk_host_name_templates
```

```
#Constructing four out-of-box bulk host name templates
my $four_octets = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "Four Octets",
    template_format => '-$1-$2-$3-$4'
);
my $three_octets = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "Three Octets",
    template_format => '-$2-$3-$4'
);
my $two_octets = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "Two Octets",
    template_format => '-$3-$4'
);
my $one_octet = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "One Octet",
    template_format => '-$4'
);

#Creating a new bulk host name templates.
my $custom_template = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "Custom Template",
    template_format => '-#1-#2-#3-#4'
);

$grid_dns->bulk_host_name_templates([$custom_template, $four_octets, $three_octets, $two_octets,
$one_octet]);
```

default_ttl( )

Use this method to set or retrieve the default TTL value of a grid DNS object. This interval tells the secondary how long the data can be cached.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number for the TTL, in seconds. *Default* value is 1/2 of lease time or 28800 seconds.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the default TTL
my $default_ttl = $grid_dns->default_ttl();
#Modify the default TTL
$grid_dns->default_ttl(28800);
```

dns64_groups( )

Use this method to set or retrieve the list of DNS64 synthesis groups.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::DNS::DNS64Group objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the list of DNS64 synthesis groups
my $dns64_groups_list = $grid_dns->dns64_groups();

#Modify it
my $group = $session->get(
                        object => "Infoblox::Grid::DNS::DNS64Group"),
                        name => "groupname",
                        prefix => "groupprefix",
                      );
$grid_dns->dns64_groups([$group]);
```

dnssec_enabled( )

Use this method to enable/disable the DNS security extension.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to enable the DNS security extension. Set the parameter to "false" to disable the feature.

The default value of the parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_enabled
my $dnssec_enabled = $grid_dns->dnssec_enabled();
#Modify dnssec_enabled
$grid_dns->dnssec_enabled("true");
```

dnssec_expired_signatures_enabled( )

Use this method to control whether the DNS grid object accepts expired signatures.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to accept expired signatures. Set the parameter to "false" to reject expired signatures.

The default value of the parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_expired_signatures_enabled
my $dnssec_expired_signatures_enabled = $grid_dns->dnssec_expired_signatures_enabled();
#Modify dnssec_expired_signatures_enabled
$grid_dns->dnssec_expired_signatures_enabled("true");
```

dnssec_ksk_algorithm()

Use this method to set or retrieve the DNSSEC key-signing key algorithm of all the zones in the grid.

This property determines which algorithm to use when generating the keys for signing the DNSKEY records.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Currently there are two major versions of the DNSSEC zone encryption mechanism. The difference between the two is in their implementation of the Next Secure (NSEC) records. The older version, which is referred to as NSEC, is based on RFCs 4033-4035. The newer version, which is referred to as NSEC3, is described in RFC 5155. The algorithm field allows you to specify which encryption version to use as well as which encryption algorithm to chose inside the defined NSEC type. Both the ZSK and KSK algorithms should use the same NSEC version; however, they are allowed to use different encryption algorithms within the same NSEC version.

The parameter can be specified as an integer or as a mnemonic. The following table lists the available algorithm values supported by the appliance, their mnemonics, and NSEC type.

```
   Value  Algorithm        Mnemonic
 NSEC algorithms:
     1     RSA/MD5         RSAMD5
     3     DSA/SHA-1       DSA
     5     RSA/SHA-1       RSASHA1
 NSEC3 algorithms:
     6     DSA/SHA-1       NSEC3DSA
     7     RSA/SHA-1       NSEC3RSASHA1
```

The default value of dnssec_ksk_algorithm parameter is 'RSASHA1'.

For algorithms not listed above, only the mnemonic representation is supported.

The full list of supported values is:
'RSAMD5','DSA','RSASHA1','NSEC3DSA','NSEC3RSASHA1','RSASHA256','RSASHA512','NSEC3RSASHA256','NSEC3RSASHA512',1,3,5,6,7.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting algorithm
my $algorithm = $grid_dns->dnssec_ksk_algorithm();

# Modifying algorithm
$grid_dns->dnssec_ksk_algorithm("DSA");
```

dnssec_ksk_rollover_interval()

Use this method to set or retrieve the rollover interval of the KSK keys.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A positive integer that specifies the number of seconds between KSK rollovers. Normally, this parameter should be larger than the ZSK rollover interval. The DNSSEC key-signing key rollover interval (in seconds) must be between 1 day and seconds to Jan 2, 2038.

The default value of the dnssec_ksk_rollover_interval parameter is 31536000 (1 year).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_ksk_rollover_interval
my $ksk_rollover = $grid_dns->dnssec_ksk_rollover_interval();

# Modifying dnssec_ksk_rollover_interval
$grid_dns->dnssec_ksk_rollover_interval(86400);
```

dnssec_ksk_size()

Use this method to set or retrieve the length of the key-signing key.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DNSSEC key-signing key possible size depends on the algorithm. Use the following table to determine the possible values for key sizes:

```
    Algorithm     Key size     Additional restrictions
  NSEC
    RSA/MD5       512-4096
    DSA/SHA1      512-1024     Key size must be divisible by 64
    RSA/SHA1      512-4096
    RSA256/SHA2   512-4096
    RSA512/SHA2   1024-4096
  NSEC3
    DSA/SHA1      512-1024     Key size must be divisible by 64
    RSA/SHA1      512-4096
    RSA256/SHA2   512-4096
    RSA512/SHA2   1024-4096
```

The default value of dnssec_ksk_size parameter is 2048.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_ksk_size
my $ksk_size = $grid_dns->dnssec_ksk_size();

# Modifying dnssec_ksk_size
$grid_dns->dnssec_ksk_size(4096);
```

dnssec_signature_expiration( )

Use this method to set or retrieve the expiration time for DNSSEC signatures.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The signature expiration time in seconds. The minimum value is 86400 (1 day), the maximum value is the number of seconds from the time you set it to January 2038.

The default value of dnssec_signature_expiration parameter is 2592000 seconds (i.e. 30 days).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_signature_expiration
my $signature_expiration = $grid_dns->dnssec_signature_expiration();

# Modifying dnssec_signature_expiration
$grid_dns->dnssec_signature_expiration(31536000);
```

dnssec_validation_enabled( )

Use this method to enable and disable DNS security validation (for the grid appliances working as clients of secure zones).

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to enable the validation. Set to "false" to disable the validation.

The default value of the parameter is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_validation_enabled
my $dnssec_validation_enabled = $grid_dns->dnssec_validation_enabled();
#Modify dnssec_validation_enabled
$grid_dns->dnssec_validation_enabled("false");
```

dnssec_trusted_keys( )

Use this method to specify the list of trusted keys for the DNSSEC feature.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference that contains a list of Infoblox::DNS::DnssecTrustedKey objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns a reference to a list of the trusted keys.

Example

```
#Get dnssec_trusted_keys
my $dnssec_trusted_keys = $grid_dns->dnssec_trusted_keys();
#Modify dnssec_trusted_keys
my $key1=Infoblox::DNS::DnssecTrustedKey->new(
   fqdn => "myzone.com",
   algorithm => "RSAMD5",
   key       => "c2ltcGxlIGtleQo="
   );
$grid_dns->dnssec_trusted_keys([$key1]);
```

dnssec_zsk_algorithm()

Use this method to set or retrieve the DNSSEC zone-signing key algorithm of all the zones in the grid.

This property determines which algorithm will be used to sign zone data.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Currently there are two major versions of the DNSSEC zone encryption mechanism. The difference between the two is in their implementation of the Next Secure (NSEC) records. The older version, which is referred to as NSEC, is based on RFCs 4033-4035. The newer version, which is referred to as NSEC3, is described in RFC 5155. The algorithm field allows you to specify which encryption version to use as well as which encryption algorithm to chose inside the defined NSEC type. Both the ZSK and KSK algorithms should use the same NSEC version; however, they are allowed to use different encryption algorithms within the same NSEC version.

The parameter can be specified as an integer or as a mnemonic. The following table lists the available algorithm values supported by the appliance, their mnemonics, and NSEC type.

```
    Value  Algorithm        Mnemonic
  NSEC algorithms:
      1     RSA/MD5         RSAMD5
      3     DSA/SHA-1       DSA
      5     RSA/SHA-1       RSASHA1
  NSEC3 algorithms:
      6     DSA/SHA-1       NSEC3DSA
      7     RSA/SHA-1       NSEC3RSASHA1
```

The default value of the dnssec_zsk_algorithm parameter is "RSASHA1".

For algorithms not listed above, only the mnemonic representation is supported.

The full list of supported values is:
'RSAMD5','DSA','RSASHA1','NSEC3DSA','NSEC3RSASHA1','RSASHA256','RSASHA512','NSEC3RSASHA256','NSEC3RSASHA512',1,3,5,6,7.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting algorithm
my $algorithm = $grid_dns->dnssec_zsk_algorithm();

# Modifying algorithm
$grid_dns->dnssec_zsk_algorithm("DSA");
```

dnssec_zsk_rollover_interval()

Use this method to set or retrieve the zone-signing key rollover interval.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A positive integer that specifies the number of seconds between ZSK rollovers. Normally, this parameter should be smaller than the KSK rollover interval. The DNSSEC zone-signing key over interval (in seconds) must be between 1 day and the number of seconds from present to Jan 2038.

The default value of the dnssec_zsk_rollover_interval parameter is 2592000 seconds (30 days).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_zsk_rollover_interval
my $zsk_rollover = $grid_dns->dnssec_zsk_rollover_interval();

# Modifying dnssec_ksk_rollover_interval
$grid_dns->dnssec_zsk_rollover_interval(86400);
```

dnssec_zsk_size()

Use this method to set or retrieve the length of zone-signing key.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The DNSSEC zone-signing key size depends on the algorithm. Use the following table to determine the possible values for key sizes:

```
  Algorithm    Key size     Additional restrictions
NSEC
  RSA/MD5      512-4096
  DSA/SHA1     512-1024     Key size must be divisible by 64
  RSA/SHA1     512-4096
  RSA256/SHA2  512-4096
  RSA512/SHA2  1024-4096
NSEC3
  DSA/SHA1     512-1024     Key size must be divisible by 64
  RSA/SHA1     512-4096
  RSA256/SHA2  512-4096
  RSA512/SHA2  1024-4096
```

The default value of the dnssec_zsk_size parameter is 1024 bits.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting dnssec_zsk_size
my $zsk_size = $grid_dns->dnssec_zsk_size();

# Modifying dnssec_zsk_size
$grid_dns->dnssec_zsk_size(4096);
```

email( )

Use this method to set or retrieve the email address of a grid DNS object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

String with email address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get email address
my $email = $grid_dns->email();
#Modify email address
$grid_dns->email("admin\@infoblox.com");
```

enable_blackhole( )

Use this method to set or retrieve the enable_blackhole flag of a grid DNS object. When enabled, the Infoblox appliance does not respond to DNS queries from clients that are in the configured blackhole_list.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_blackhole flag or "false" to deactivate and unset it. The default value is "false".

Returns

When you specify a parameter and the modification succeeds, the method returns "true". When the operation fails, the method returns "false".

When you do not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_blackhole flag
my $enable_blackhole = $grid_dns->enable_blackhole();
#Modify enable_blackhole recursive query flag
$grid_dns->enable_blackhole("true");
```

enable_blacklist( )

Use this method to enable or disable a blacklist globaly.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_blacklist flag or "false" to deactivate/unset it. The default is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_blacklist
my $enable_blacklist = $grid_dns->enable_blacklist( );
#Modifying enable_blacklist
$grid_dns->enable_blacklist("true");
```

enable_dns64( )

Use this method to enable/disable DNS64 support.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_dns64 flag or "false" to deactivate and unset it. The default value is "false".

Returns

When you specify a parameter and the modification succeeds, the method returns "true". When the operation fails, the method returns "false".

When you do not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dns64 flag
my $enable_dns64 = $grid_dns->enable_dns64();
#Modify enable_dns64 recursive query flag
$grid_dns->enable_dns64("true");
```

enable_host_rrset_order( )

Use this method to set or retrieve the grid-level enable_host_rrset_order value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_host_rrset_order flag or "false" to deactivate and unset it. The default value is "false".

Returns

When you specify a parameter and the modification succeeds, the method returns "true". When the operation fails, the method returns "false".

When you do not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_host_rrset_order
my $host_rrset_order = $grid_dns->enable_host_rrset_order();
# Set enable_host_rrset_order
$grid_dns->enable_host_rrset_order("true");

=back
```

expire_after( )

Use this method to set or retrieve the expiration time of a grid DNS object. If the secondary DNS server fails to contact the primary server for the specified interval, the secondary server stops giving out answers about the zone because the zone data is too old to be useful.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number with the seconds for the expiration time. *Default* value is 2419200 seconds (4 weeks).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the expiration time
my $expire_after = $grid_dns->expire_after();
#Modify the expiration time
$grid_dns->expire_after(2419200);
```

forward_only( )

Use this method to set or retrieve the forward only flag of a grid DNS object. On enabling, the Infoblox appliance will send queries to forwarders only, and not to internal or Internet root servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the forward_only flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the forward only flag
my $forward_only = $grid_dns->forward_only();
#Modify the forward only
$grid_dns->forward_only("true");
```

forward_updates( )

Use this method to set or retrieve the forward_updates flag of a grid DNS object. This enables a secondary server to receive updates from the DHCP server, and then forward them to the primary DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the forward_updates flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the forward updates flag
my $forward_updates = $grid_dns->forward_updates();
#Modify the forward updates flag
$grid_dns->forward_updates("true");
```

forwarders( )

Use this method to set or retrieve the forwarders list of a grid DNS object. A forwarder is essentially a name server to which other name servers first send all of their off-site queries. The forwarder builds up a cache of information, avoiding the necessity for the other name servers to send queries off-site.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IP address(es).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the forwarders list
my $forwarders_list = $grid_dns->forwarders();
#Modify the forwarders list
$grid_dns->forwarders(["2.2.2.2", "3.3.3.3"]);
```

host_name_restriction_policy( )

Use this method to set or retrieve the host name restriction policy of a grid DNS object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

String with the name of predefined record policies or user-defined record policies. The default value is the predefined record policy "Allow Underscore".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get host name restriction policy
my $host_name_restriction_policy = $grid_dns->host_name_restriction_policy();
#Modify host name restriction policy
$grid_dns->host_name_restriction_policy("Allow Any");
```

logging_categories( )

Use this method to set or retrieve the logging categories. It is possible to specify 14 BIND logging message categories to be captured by syslog.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains one or more of the following values:

"general" - Records the BIND messages that are not specifically classified.

"config" - Records the configuration file parsing messages.

"dnssec" - Records the DNSSEC-signed responses.

"network" - Records the network operation messages.

"queries" - Records the query response messages.

"security" - Records the approved and denied requests.

"xfer-in" - Records zone transfer messages from the remote name servers to the appliance.

"xfer-out" - Records zone transfer messages from the Infoblox appliance to remote name servers.

"update" - Records the dynamic update instances.

"resolver" - Records the DNS resolution instances, including recursive queries from resolvers.

"notify" - Records the asynchronous zone change notification messages.

"lame_servers" - Records bad delegation instances.

"database" - Records BIND internal database processes.

"client" - Records client requests.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the logging categories list
my $logging_categories_list = $grid_dns->logging_categories();
#Modify the logging categories list
$grid_dns->logging_categories(["general", "database", "queries"]);
```

logging_facility( )

Use this method to set or retrieve the logging facility of a grid DNS object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the facility name. Valid values are: "daemon", "local0", "local1", "local2", "local3", "local4", "local5", "local6", and "local7".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the logging facility
my $logging_facility = $grid_dns->logging_facility();
#Modify the logging facility
$grid_dns->logging_facility("daemon");
```

member_secondary_notify( )

Use this method to set or retrieve the member_secondary_notify flag of a grid DNS object. When enabled, this feature allows grid members that are authoritative secondary servers to send notification messages to external name servers, if the grid member that is primary for a zone fails or loses connectivity.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set member_secondary_notify flag or "false" to deactivate/unset it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the member secondary notification flag
my $member_secondary_notify = $grid_dns->member_secondary_notify();
#Modify the member secondary notification flag
$grid_dns->member_secondary_notify("false");
```

negative_ttl( )

Use this method to set or retrieve the negative TTL value of a grid DNS object. This interval tells the secondary how long data can be cached for "Does Not Respond" responses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number for the negative TTL, in seconds. *Default* value is 900 seconds.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the negative ttl value
my $negative_ttl = $grid_dns->negative_ttl();
#Modify the negative ttl value
$grid_dns->negative_ttl(800);
```

notify_delay( )

Use this method to specify with how many seconds of delay the notify messages are sent to secondaries.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are unsigned integer between 5 and 86400, inclusive. Default value is 5.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting notify_delay
my $notify_delay = $griddns->notify_delay( );
#Modifying notify_delay
$griddns->notify_delay(5);
```

nxdomain_redirect( )

Use this method to enable or disable NXDOMAIN redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the nxdomain_redirect flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting nxdomain_redirect
my $nxdomain_redirect = $griddns->nxdomain_redirect( );
#Modifying nxdomain_redirect
$griddns->nxdomain_redirect("true");
```

nxdomain_redirect_addresses( )

Use this method to set the NXDOMAIN redirection addresses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array of IPv4 addresses. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_redirect_addresses
my $nxdomain_redirect_addresses = $griddns->nxdomain_redirect_addresses( );
#Modify nxdomain_redirect_addresses
$griddns->nxdomain_redirect_addresses(["10.0.0.10", "10.0.0.20"]);
```

nxdomain_redirect_ttl( )

Use this method to set or retrieve the TTL value of synthetic DNS responses that result from NXDOMAIN redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer, from 0 to 4294967295, that represents the TTL in seconds. The default value is 60.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_redirect_ttl
my $nxdomain_redirect_ttl = $griddns->nxdomain_redirect_ttl( );
```

```
#Modify nxdomain_redirect_ttl
$griddns->nxdomain_redirect_ttl(60);
```

nxdomain_log_query( )

Use this method to set or retrieve the flag that indicates whether NXDOMAIN redirection queries are logged.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable logging, or "false" to deactivate it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_log_query
my $nxdomain_log_query = $griddns->nxdomain_log_query( );
#Modify nxdomain_log_query
$griddns->nxdomain_log_query("true");
```

nxdomain_rulesets( )

Use this method to specify or retrieve Infoblox::DNS::Ruleset object names assigned at the grid level for NXDOMAIN redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains the names of the Infoblox::DNS::Ruleset objects assigned to the grid. The names must be in string format. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_rulesets
my $nxdomain_rulesets = $griddns->nxdomain_rulesets();
#Modify nxdomain_rulesets
$griddns->nxdomain_rulesets(["ruleset1", "ruleset2"]);
```

notify_source_port( )

Use this method to set or retrieve the source port number for notify messages. When requesting zone transfers from the primary server, some secondary DNS servers use the source port number (the primary server used to send the notify message) as the destination port number in the zone transfer request.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Source port number for notify messages. Valid values are between 1 and 63999. *Default* is picked by BIND.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the notify source port value
my $notify_source_port = $grid_dns->notify_source_port();
#Modify the notify source port value
$grid_dns->notify_source_port(77);
```

nsgroup_default( )

Use this method to set or retrieve the default nameserver group.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name of the default nameserver group. *Default* value is None.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the default nameserver group
my $nsgroup_default = $grid_dns->nsgroup_default();
#Modify the default nameserver group
$grid_dns->nsgroup_default("ns_group_A");
```

nsgroups( )

Use this method to set or retrieve the list of nameserver groups. A name server group is a collection of one primary DNS server and one or more secondary DNS servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::DNS::Nsgroup and Infoblox::DNS::Member object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the nameserver group
my $nsgroup = $grid_dns->nsgroups();
#Modify the nameserver group
#Creating Member objects.
my $member1 = Infoblox::DNS::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
    stealth  => "false"
    );

#Creating a nameserver object.
my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name     => "ns1.test.extra",
    ipv4addr => "4.4.4.4",
    stealth  => "false",
    TSIGname => "tsig_name",
    TSIGkey  => "ZX1jVJl7C58GT/sc7Q3ucA=="
    );

#Creating a nameserver object.
my $nameserver2 = Infoblox::DNS::Nameserver->new(
    name     => "ns2.test.extra",
    ipv4addr => "5.5.5.5",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
    );

my $nsg1 = Infoblox::Grid::DNS::Nsgroup->new(
    name        => "ns_group_A",
    primary     => $member1,
    secondaries => [$nameserver1, $nameserver2]
    );

$grid_dns->nsgroups([$nsg1]);
```

preserve_host_rrset_order_on_secondaries( )

Use this method to set or retrieve the grid-level preserve_host_rrset_order_on_secondaries value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the preserve_host_rrset_order_on_secondaries flag or "false" to unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get preserve_host_rrset_order_on_secondaries
my $preserve_order = $griddns->preserve_host_rrset_order_on_secondaries();
# Set preserve_host_rrset_order_on_secondaries
$griddns->preserve_host_rrset_order_on_secondaries("true");
```

query_source_port( )

Use this method to set or retrieve the query source port of a grid DNS object. Specifying a source port number for recursive queries ensures that a firewall will allow the response.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Source port number for recursive queries . Valid values are between 1 and 63999. *Default* is picked by BIND.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the query source port
my $query_source_port = $grid_dns->query_source_port();
#Modify the query source port
$grid_dns->query_source_port(99);
```

record_name_policies( )

Use this method to set or retrieve the list of record name policies.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::RecordNamePolicy object(s). By default there are three predefined policies.

Policy Name Regex Applied

"Allow Any" .+

"Allow Underscore" ^[-a-zA-Z0-9_]+$

"Strict Hostname Checking" ^[a-zA-Z0-9]$|^[a-zA-Z0-9][-a-zA-Z0-9]*[a-zA-Z0-9]$

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get record name policies
my $record_name_policies = $grid_dns->record_name_policies();
#Modify record_name_policies

#Creating record name policy objects.
my $recpolicy1 = Infoblox::Grid::RecordNamePolicy->new(
    regex => "*",
    name  => "Local Policy",
    );
my $recpolicy2 = Infoblox::Grid::RecordNamePolicy->new(
    regex => ".+",
    name  => "Allow Any",
    );
my $recpolicy3 = Infoblox::Grid::RecordNamePolicy->new(
    regex => "^[-a-zA-Z0-9_]+\$",
```

```
        name  => "Allow Underscore",
        );
    my $recpolicy4 = Infoblox::Grid::RecordNamePolicy->new(
        regex => "^[a-zA-Z0-9]\$|^[a-zA-Z0-9][-a-zA-Z0-9]*[a-zA-Z0-9]\$",
        name  => "Strict Hostname Checking",
        );

    $grid_dns->record_name_policies([$recpolicy1, $recpolicy2, $recpolicy3, $recpolicy4]);
```

recursive_query_list( )

Use this method to set or retrieve the recursive query list of a Grid DNS object. This list specifies from which IP addresses, networks and TSIG key authenticated clients recursive queries are allowed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IP addresses, networks, and/or Infoblox::DNS::TSIGKey objects. To deny queries, add "!" as a prefix to IP addresses and/or networks. Specify "any" to allow recursive queries from any IP address.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
    #Get the recursive query list
    my $recursive_query_list = $grid_dns->recursive_query_list();
    #Modify the recursive query list
    $grid_dns->recursive_query_list(["any"]);

    #Creating a TSIG key object
    my $tsig_key2 = $session->gen_tsig_key();

    my $key2 = Infoblox::DNS::TSIGKey->new(
        name => "rcl_key",
        key  => $tsig_key2
        );
    #Modify the recursive query list
    $grid_dns->recursive_query_list(["10.0.0.10", "20.0.0.0/24","2010::256/64", $key2,
"!","2011::23cb/64","!","30.0.0.30", "!","40.10.0.0/24"]);
    #Modify the recursive query list
    $grid_dns->recursive_query_list(undef);
```

refresh_timer( )

Use this method to set or retrieve the refresh time. This interval tells the secondary how often to send a message to the primary for a zone to check that its data is current, and retrieve fresh data if it is not.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number with the seconds for the refresh time. *Default* value is 10800 seconds (3 hours).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
    #Get the refresh time
    my $refresh_timer = $grid_dns->refresh_timer();
    #Modify the refresh time
    $grid_dns->refresh_timer(108);
```

retry_timer( )

Use this method to set or retrieve the retry time. This interval tells the secondary how long to wait before attempting to recontact the primary after a connection failure occurs between the two servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number with the seconds for the retry time. *Default* value is 3600 seconds (1 hour).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the retry time
my $retry_timer = $grid_dns->retry_timer();
#Modify the retry time
$grid_dns->retry_timer(360);
```

rootNS( )

Use this method to set or retrieve the list of customized root nameserver(s). You can use Internet root name servers or specify host names and IP addresses of custom root name servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::RootNameServer object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the root nameserver list
my $rootNS_list = $grid_dns->rootNS();
#Modify the root nameserver list
#Creating RootNameServer objects.
my $rootns1 = Infoblox::Grid::RootNameServer->new(
    ipv4addr  => "44.44.44.44",
    host_name => "test1.com",
    );

my $rootns2 = Infoblox::Grid::RootNameServer->new(
    ipv4addr  => "55.5.5.5",
    host_name => "test2.com",
    );

$grid_dns->rootNS([$rootns1, $rootns2]);
```

sortlist( )

Use this method to set or retrieve the sort list of a grid DNS object. A sort list sorts the order of addresses in responses made to DNS queries.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::DNS::Sortlist object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the sort list
my $sortlist = $grid_dns->sortlist();
#Modify the sort list
#Creating a SortList object.
my $sort1 = Infoblox::DNS::Sortlist->new(
    match_list      => ["10.20.1.0/24", "10.20.2.0/24"],
    source_ipv4addr => "1.2.3.4"
    );

$grid_dns->sortlist([$sort1]);
```

transfer_excluded_servers( )

Use this method to set or retrieve the list of excluded servers for zone transfers of a grid DNS object. This list exclude the specified DNS servers during zone transfers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IP addresses of DNS servers.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the transfer excluded server list
my $transfer_excluded_servers = $grid_dns->transfer_excluded_servers();
#Modify the transfer excluded server list
$grid_dns->transfer_excluded_servers(["10.0.0.10", "20.0.0.0"]);
```

transfer_format( )

Use this method to set or retrieve the zone transfer format. The zone transfer format determines the BIND format for a zone transfer. This provides tracking capabilities for single or multiple transfers and their associated servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is one of the following:

"many_answers" - Includes as many records as the packet size allows.

"one_answer" - Includes one record per packet.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the zone transfer format
my $transfer_format = $grid_dns->transfer_format();
#Modify the zone transfer format
$grid_dns->transfer_format("many_answers");
```

zone_deletion_double_confirm( )

Use this method to enable or disable double confirmation during zone deletion.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the zone_deletion_double_confirm flag or "false" to deactivate/unset it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the double confirmation of zone deletion flag
my $zone_deletion_double_confirm = $grid_dns->zone_deletion_double_confirm();
#Modify the double confirmation of zone deletion flag
$grid_dns->zone_deletion_double_confirm("false");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as get and modify. This sample code also includes error handling for the operations.

#Preparation prior to getting and modifying a grid DNS object

```
use strict;
use Infoblox;

#Creating a session to appliance.
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
    );
unless( $session ){
    die("Constructor for session failed:",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print" Session object created successfully \n";

#Creating RootNameServer objects.
my $rootns1 = Infoblox::Grid::RootNameServer->new(
    ipv4addr  => "44.44.44.44",
    host_name => "test1.com",
    );
unless( $rootns1 ){
    die("Constructor for rootnameserver1 failed:",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print" Rootnameserver1 object created successfully \n";

my $rootns2 = Infoblox::Grid::RootNameServer->new(
    ipv4addr  => "55.5.5.5",
    host_name => "test2.com",
    );
unless( $rootns2 ){
    die("Constructor for rootnameserver2 failed:",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print" Rootnameserver2 object created successfully \n";

#Creating Transfer Key objects.
my $key1 = Infoblox::DNS::TSIGKey->new(
    name => "tsig_xfer",
    key  => ":2xCOMPAT"
    );
unless( $key1 ){
    die("Constructor for transfer key \(key1\) failed:",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print" Transfer key object \(key1\) created successfully \n";

my $tsig_key1 = $session->gen_tsig_key();
unless( $tsig_key1 ){
    die("Generate tsig_key1 failed: ",
    $session->status_code() . ":" . $session->status_detail());
}

my $key2 = Infoblox::DNS::TSIGKey->new(
    name => "transfer_key",
    key  => $tsig_key1
    );
unless( $key2 ){
    die("Constructor for transfer key \(key2\) failed:",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print" Transfer key object \(key2\) created successfully \n";

#Creating Update Key object.
my $update_key1 = $session->gen_tsig_key();
unless( $update_key1 ){
    die("Generate update_key1 failed:",
    $session->status_code() . ":" . $session->status_detail());
}

my $up_key1 = Infoblox::DNS::TSIGKey->new(
    name => "update_key",
    key  => $update_key1
    );
unless( $up_key1 ){
    die("Constructor for update key failed: ",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
```

```
print" Update key object created successfully \n";

#Creating a SortList object.
my $sort1 = Infoblox::DNS::Sortlist->new(
    match_list      => ["10.20.1.0/24", "10.20.2.0/24"],
    source_ipv4addr => "1.2.3.4"
    );
unless( $sort1 ){
    die("Constructor for sortlist failed: ",
    Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print" Sortlist object created successfully \n";
```

#Get and modify the DNS object

```
#Getting the DNS object from appliance through session.
my @result = $session->get(
    object => "Infoblox::Grid::DNS",
    name   => "Infoblox"
    );
unless( @result ){
    die("Get grid DNS failed: ",
    $session->status_code() . ":" . $session->status_detail());
}
print" Get grid DNS object found at least 1 matching entry \n";

my $grid_dns = $result[0];
if ($grid_dns) {

#Modifying the value of the specified object
   $grid_dns->allow_query(["10.0.0.10", "20.0.0.0/24", "!" . "30.0.0.30", "!" . "40.10.0.0/24"]);
   $grid_dns->allow_recursive_query("true");
   $grid_dns->allow_transfer(["10.0.0.10", "20.0.0.0/24", $key1, "!"."30.0.0.30", "!"."40.10.0.0/24",
$key2]);
   $grid_dns->allow_update(["10.0.0.10", "20.0.0.0/24", "!"."30.0.0.30", "!"."40.10.0.0/24", $up_key1]);
   $grid_dns->default_ttl(28800);
   $grid_dns->expire_after(2419200);
   $grid_dns->forwarders(["2.2.2.2", "3.3.3.3"]);
   $grid_dns->logging_facility("daemon");
   $grid_dns->negative_ttl(900);
   $grid_dns->recursive_query_list(["10.0.0.10", "20.0.0.0/24", "!" . "30.0.0.30", "!" . "40.10.0.0/24"]);
   $grid_dns->refresh_timer(10800);
   $grid_dns->retry_timer(3600);
   $grid_dns->rootNS([$rootns1, $rootns2]);
   $grid_dns->sortlist([$sort1]);
   $grid_dns->transfer_excluded_servers(["10.0.0.10", "20.0.0.0"]);
   $grid_dns->transfer_format("many_answers");

 #Applying the changes
 my $response = $session->modify($grid_dns);
 unless( $response ){
    die("Modify grid DNS failed: ",
    $session->status_code() . ":" . $session->status_detail());
 }
 print" Modify grid DNS successful \n";
  }
  else {
   print "No grid DNS object found.";
  }

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::Sortlist, Infoblox::DNS::TSIGKey, Infoblox::DNS::DnssecTrustedKey, Infoblox::Grid::DNS::Nsgroup, Infoblox::Grid::RootNameServer, Infoblox::DNS::Ruleset, Infoblox::Session->get(), Infoblox::Session->modify(),Infoblox::Session

COPYRIGHT

Infoblox::Grid::DNS::BulkHostNameTemplate - Manages the DNS bulk host name formats defined at the grid level.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DNS->bulk_host_name_templates( )
- METHODS
  - template_name( )
  - template_format( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::DNS::BulkHostNameTemplate - Manages the DNS bulk host name formats defined at the grid level.

DESCRIPTION

Bulk host name template defines the format based on which the host names are generated from a bulk host..

CONSTRUCTOR

```
my $bulk_host_name_template = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => $string,    #Required
    template_format => $string     #Required
);
```

MODULE METHODS

The following functions are available to apply to the bulk host name template object.

Infoblox::Grid::DNS->bulk_host_name_templates( )

Use this function to specify a bulk host name template at the grid level in the Infoblox appliance.

Example

```
#Create a BulkHostNameTemplate object.
my $bulk_host_name_template = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
   template_name   => "test",
   template_format => "-#1-#2-#3-#4",
);

#Configure bulk host name template on the Infoblox grid object
my $response = $grid_dns->bulk_host_name_templates([ $test, $four_octets, $three_octets,
$two_octets, $one_octet ] );
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attributes values of a bulk host name template object.

template_name( )

Use this method to set or retrieve the name of bulk host name template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

String with the name of the bulk host name template.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get template_name
my $template_name = $bulk_host_name_template->template_name();
#Modify template_name
$template_name->name("new_name");
```

template_format( )

Use this method to set or retrieve the format of bulk host name template.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The format of bulk host name template. It should follow certain rules (please use Administration Guide as reference).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get template_format
my $template_format = $bulk_host_name_template->template_format();
#Modify template_format
$bulk_host_name_template>template_format("-#1-#2-#3-#4");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to a bulk host name template object such as create, modify, and remove. Also, this sample code includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
    die( "Construct session failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail() );
}
print "Session created successfully\n";

#First construct four out-of-box bulk host name template objects

my $four_octets = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "Four Octets",
    template_format => '-$1-$2-$3-$4'
);
unless ($four_octets) {
    die(
        "Construct bulk host name template object four_octets failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail()
    );
}
print "Bulk host name template object four_octets created successfully\n";

my $three_octets = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "Three Octets",
    template_format => '-$2-$3-$4'
);
unless ($three_octets) {
    die(
        "Construct bulk host name template object three_octets failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail()
```

```
    );
}
print "Bulk host name template object three_octets created successfully\n";

my $two_octets = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "Two Octets",
    template_format => '-$3-$4'
);
unless ($two_octets) {
    die(
        "Construct bulk host name template object two_octets failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail()
    );
}
print "Bulk host name template object two_octets created successfully\n";

my $one_octet = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "One Octet",
    template_format => '-$4'
);
unless ($one_octet) {
    die(
        "Construct bulk host name template object one_octet failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail()
    );
}
print "Bulk host name template object one_octet created successfully\n";

#Create a new bulk host name template object

my $my_template = Infoblox::Grid::DNS::BulkHostNameTemplate->new(
    template_name   => "My Template",
    template_format => "#4",
);
unless ($my_template) {
    die(
        "Construct bulk host name template object my_template failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail()
    );
}
print "Bulk host name template object my_template created successfully\n";

#Get the Member DNS object to add a bulk host name template object to it

my $grid_dns =
  $session->get( "object" => "Infoblox::Grid::DNS", "grid" => "Infoblox" );
unless ($grid_dns) {
    die( "Get Grid DNS failed: ",
        $session->status_code() . ":" . $session->status_detail() );
}

$grid_dns->bulk_host_name_templates(
    [ $my_template, $four_octets, $three_octets, $two_octets, $one_octet ] );

$session->modify($grid_dns)
  or die( "modify Grid DNS failed:",
    $session->status_code(), $session->status_detail() );
print "Modify grid DNS successful for bulk host name template object\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::DNS, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::DNS::DNS64Group - DNS64 Synthesis Group object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - clients( )
  - disable( )
  - exclude( )
  - extensible_attributes( )
  - name( )
  - mapped( )
  - prefix( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::DNS::DNS64Group - DNS64 Synthesis Group object

DESCRIPTION

A DNS64 synthesis group is used to configure the DNS64 feature.

CONSTRUCTOR

```
 my $group = Infoblox::Grid::DNS::DNS64Group->new(
     name                  => $string,
#Required
     prefix                => $ipv6prefix,
#Required
     clients               => [ $ipv4addr | $ipv4_subnet/prefix | $ipv6addr | $ipv6_subnet/prefix ,... ],
#Optional / Default is an empty list
     comment               => $string,
#Optional
     disable               => "true" | "false"
#Optional / Default is "false"
     exclude               => [ $ipv4addr | $ipv4_subnet/prefix | $ipv6addr | $ipv6_subnet/prefix ,... ],
#Optional / Default is an empty list
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     mapped                => [ $ipv4addr | $ipv4_subnet/prefix | $ipv6addr | $ipv6_subnet/prefix ,... ],
#Optional / Default is an empty list
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a DNS64 synthesis group object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $group = Infoblox::Grid::DNS::DNS64Group->new(
    name        => "dns64group_A",
    prefix      => "64:FF9B::/96",
    );
```

```
#Submit for addition
my $response = $session->add( $group );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific DNS64 synthesis group object:

 name                 - Optional. Group name in string format.
 prefix               - Optional. An IPv6 prefix.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 comment              - Optional. User-specified comment for the group.
```

Examples

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS::DNS64Group",
    name   => "dns64group_A",
    );

my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS::DNS64Group",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use this method to modify the comment.
$group->comment("this is a modified comment");
#Submit modification
my $response = $session->modify( $group );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS::DNS64Group",
    name   => "dns64group_A",
     );
#Find the desired object on the retrieved list
my $desired_nsgroup= $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_nsgroup );
```

Infoblox::Session->search( )

Use this method to search for DNS64 synthesis group objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a DNS64 synthesis group object:

 name                 - Optional. Group name in string format. (regular expression)
 prefix               - Optional. An IPv6 prefix. (regular expression)
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 comment              - Optional. User-specified comment for the group. (regular expression)
```

Examples

```
# Search for all DNS64 synthesis group objects
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::DNS::DNS64Group",
    name   => ".*",
    );
```

```
# Search for all DNS64 synthesis group objects with "Santa Clara" as the value for the "Site" extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS::DNS64Group",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a DNS64 synthesis group object.

comment( )

Use this method to set or retrieve the comment field of the DNS64 synthesis group object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Comment of the group in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting comment
my $comment = $group->comment();
#Modifying comment
$group->comment("Some comment");
```

clients( )

Use this method to set or retrieve the clients ACL list of a DNS64 synthesis roup object. This is the list of clients for which synthesized records are returned. The default is "any".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IPv4 or IPv6 addresses and networks. To exclude addresses and networks, add "!" as a prefix to the IP addresses and networks. The default value is an empty list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting clients
my $clients = $group->clients();
#Modifying clients
$group->clients(["2001::333/64", "!2001::1234", "20.0.0.0/8"]);
```

disable( )

Use this method to set or retrieve the disable flag of a DNS64 synthesis group object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
```

```
my $disable = $fixed_addr->disable();
#Modify disable
$fixed_addr->disable("true");
```

exclude( )

Use this method to set or retrieve the exclude ACL list of a DNS64 synthesis group object. AAAA records which match this ACL are treated as non-existent for the purposes of synthesis. By default, the ACL is empty and no records match.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IPv4 or IPv6 addresses and networks. To exclude addresses and networks, add "!" as a prefix to the IP addresses and networks. The default value is an empty list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting exclude
my $exclude = $group->exclude();
#Modifying exclude
$group->exclude(["2001::333/64", "!2001::1234"]);
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a DNS64 synthesis group object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $group->extensible_attributes();
#Modify extensible attributes
$group->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the DNS64 synthesis group name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the group in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting name
my $name = $group->name();
#Modifying name
$group->name("ns_group_2");
```

mapped( )

Use this method to set or retrieve the mapped ACL list of a DNS64 synthesis group object. The appliance generates AAAA records only for A records that match this ACL. If the ACL is empty, as it is by default, then all records will match.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IPv4 or IPv6 addresses and networks. To deny addresses and networks, add "!" as a prefix to the IP addresses and networks. The default value is an empty list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting mapped
my $mapped = $group->mapped();
#Modifying mapped
$group->mapped(["40.10.0.0/24", "!40.10.3.4]);
```

prefix( )

Use this method to set or retrieve the DNS64 synthesis group prefix. The prefix is an IPv6 subnet prefix with a length of /32, /40, /48, /56, /64, or /96.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Prefix of the group in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting prefix
my $prefix = $group->prefix();
#Modifying prefix
$group->prefix("64:FF9B::/96");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as modify and remove. This sample also includes error handling for the operations.

#Preparation prior to the addition of a DNS64 synthesis group object

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device
my $session = Infoblox::Session->new(
                                     master   => "192.168.1.2",
                                     username => "admin",
                                     password => "infoblox"
                                    );
unless ($session) {
    die("Construct session failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a DNS64 synthesis group object and add it to the appliance

```
my $group = Infoblox::Grid::DNS::DNS64Group->new(
                                                 name => 'testgroup',
```

```
                                                    prefix => '64:FF9B::/96',
                                                    clients => ["2001::333/64", "!2001::1234", "20.0.0.0/8"],
                                                    comment => 'some comment',
                                                   );

unless ($group) {
    die("Construct DNS64 synthesis group failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DNS64 synthesis group object created successfully\n";

my $response = $session->add( $group );
unless ($response) {
    die("Adding DNS64 synthesis group failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Add DNS64 synthesis group success.\n" if ( $response );

#Add the DNS64 synthesis group object to the grid and enable it

my @result_array = $session->get( "object" => "Infoblox::Grid::DNS" ,  "grid" => "Infoblox" );

if ( defined @result_array  and scalar( @result_array ) > 0 ) {
    my $object = $result_array[0];
    if ( ref( $object ) eq "Infoblox::Grid::DNS" ) {
        $object->dns64_groups( [ $group ] );
        $object->enable_dns64("true");

        $response = $session->modify( $object );
        print "Add DNS64 synthesis group to grid dns properties success.\n" if ( $response );
             }

    unless ($object) {
        die("Add DNS64 synthesis group to grid dns properties failed: ",
            $session->status_code() . ":" . $session->status_detail());
    }
     }

#Modify the DNS64 synthesis group object

@result_array = $session->get( "object" => "Infoblox::Grid::DNS::DNS64Group" ,  name => "testgroup" );

if ( defined @result_array  and scalar( @result_array ) > 0 ) {
    my $object = $result_array[0];
    if ( ref( $object ) eq "Infoblox::Grid::DNS::DNS64Group" ) {

        $object->clients(["2001::333/64", "!2001::1234"]);

        $response = $session->modify( $object );
        print "Modify DNS64 synthesis group success.\n" if ( $response );
             }

    unless ($object) {
        die("Modify Group failed: ",
            $session->status_code() . ":" . $session->status_detail());
    }
     }

#Remove a DNS64 synthesis group object

#First let's remove it from the grid configuration

@result_array = $session->get( "object" => "Infoblox::Grid::DNS" ,  "grid" => "Infoblox" );

if ( defined @result_array  and scalar( @result_array ) > 0 ) {
    my $object = $result_array[0];
    if ( ref( $object ) eq "Infoblox::Grid::DNS" ) {

        $object->dns64_groups( [ ] );
        $object->enable_dns64( "false" );

        $response = $session->modify( $object );
        print "Remove DNS64 synthesis group from the grid success.\n" if ( $response );
             }

    unless ($object) {
```

```
        die("Remove DNS64 synthesis group failed: ",
            $session->status_code() . ":" . $session->status_detail());
    }
     }
```

#And now remove the group

```
@result_array = $session->get( "object" => "Infoblox::Grid::DNS::DNS64Group" ,  name => "testgroup" );

if ( defined @result_array  and scalar( @result_array ) > 0 ) {
    my $object = $result_array[0];
    if ( ref( $object ) eq "Infoblox::Grid::DNS::DNS64Group" ) {

        $response = $session->remove( $object );
        print "Remove DNS64 synthesis group success.\n" if ( $response );
             }

    unless ($object) {
        die("Remove DNS64 synthesis group failed: ",
            $session->status_code() . ":" . $session->status_detail());
    }
     }

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session,Infoblox::Session->get(),Infoblox::Session->modify(),Infoblox::Session->search(),Infoblox::Session->remove()Infoblox::Grid::DNS, Infoblox::Grid::Member::DNS, Infoblox::DNS::View

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::DNS::Nsgroup - Creates a name server group.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- MODULE METHODS
  - Infoblox::Grid::DNS->nsgroups( )
- METHODS
  - name( )
  - comment( )
  - extensible_attributes( )
  - primary( )
  - ms_ad_integrated( )
  - secondaries( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::DNS::Nsgroup - Creates a name server group.

DESCRIPTION

A name server group is a collection of one primary DNS server and one or more secondary DNS servers. Grouping a commonly used set of primary and secondary DNS servers together simplifies zone creation, allowing you to specify a single name server group instead of specifying multiple name servers individually.

CONSTRUCTOR

```
 my $nsg1 = Infoblox::Grid::DNS::Nsgroup->new(
     name        => $string,                                                          #Required
     comment     => $string,                                                          #Optional
     extensible_attributes => { $string => $string|$num, $string => [ $string|$num, ...], ... },
#Optional / Default is undefined
     ms_ad_integrated  => "true" | "false"                                            #Optional
/ Default is "false"
     primary     => $Member | [$Nameserver1, $Nameserver2, $msserver1, ...],          #Optional
/ Infoblox::DNS::Member object or a list of Infoblox::DNS::Nameserver / Infoblox::DNS::MSServer object(s).
     secondaries => [$Member1, $Member2, $Nameserver1, $Nameserver2, $msserver1, ...]
#Optional / List of Infoblox::DNS::Member object(s) and/or Infoblox::DNS::Nameserver and/or
Infoblox::DNS::MSServer object(s).
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to an Nsgroup object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Creating Member objects.
my $member1 = Infoblox::DNS::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
    stealth  => "false"
    );

my $member2 = Infoblox::DNS::Member->new(
```

```
    name     => "member2.test.com",
    ipv4addr => "2.2.2.2",
    stealth  => "false"
    );

#Creating a nameserver object.
my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name     => "ns1.test.extra",
    ipv4addr => "4.4.4.4",
    stealth  => "false",
    TSIGname => "tsig_name",
    TSIGkey  => "ZX1jVJl7C58GT/sc7Q3ucA=="
    );

#Creating a nameserver object.
my $nameserver2 = Infoblox::DNS::Nameserver->new(
    name     => "ns2.test.extra",
    ipv4addr => "5.5.5.5",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
    );

#Creating a nameserver object.
my $nameserver3 = Infoblox::DNS::Nameserver->new(
    name     => "ns3.test.extra",
    ipv6addr => "12ab::34a",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
    );

my $dnsmsserver = Infoblox::DNS::MSServer->new(
                                          address     => "10.0.0.1",
                                         );

#Construct an object
my $nsg1 = Infoblox::Grid::DNS::Nsgroup->new(
    name        => "ns_group_A",
    primary     => $member1,
    secondaries => [$member2, $nameserver1, $nameserver2,$nameserver3, $dnsmsserver]
    );

#Submit for addition
my $response = $session->add( $nsg1 );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific Nsgroup object:

 name - Required. A name server group name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS::Nsgroup",
    name   => "ns_group_A",
    );

my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS::Nsgroup",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use this method to modify the comment.
$nsg1->comment("this is a modified comment");
```

```
#Submit modification
my $response = $session->modify( $nsg1 );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS::Nsgroup",
    name   => "ns_group_A",
     );
#Find the desired object from the retrieved list.
my $desired_nsgroup= $retrieved_objs[0];
#Submit for removal
my $response = $session->remove( $desired_nsgroup );
```

Infoblox::Session->search( )

Use this method to search for Nsgroup objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for an Nsgroup object:

 name - Required. A host name in string format (regular expression).
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Examples

```
# Search for all Nsgroup objects
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::DNS::Nsgroup",
    name   => ".*",
    );

# search for all Nsgroup objects with "Santa Clara" as the value for the "Site" extensible
attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS::Nsgroup",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

MODULE METHODS

This section describes all the methods that you can appy to an Nsgroup object.

Infoblox::Grid::DNS->nsgroups( )

Use this function to specify a name server group at the grid level on the Infoblox appliance. See Infoblox::Grid::DNS->nsgroups() for parameters and return values.

Example

```
#Creating Member objects.

my $member1 = Infoblox::DNS::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
    stealth  => "false"
    );

my $member2 = Infoblox::DNS::Member->new(
    name     => "member2.test.com",
    ipv4addr => "2.2.2.2",
    stealth  => "false"
    );

#Creating a nameserver object.
my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name     => "ns1.test.extra",
    ipv4addr => "4.4.4.4",
    stealth  => "false",
```

```
    TSIGname => "tsig_name",
    TSIGkey  => "ZX1jVJl7C58GT/sc7Q3ucA=="
    );

#Creating a nameserver object.
my $nameserver2 = Infoblox::DNS::Nameserver->new(
    name     => "ns2.test.extra",
    ipv4addr => "5.5.5.5",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
    );

#Creating a nameserver object.
my $nameserver3 = Infoblox::DNS::Nameserver->new(
    name     => "ns3.test.extra",
    ipv6addr => "12ab::34a",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
    );

my $nsg1 = Infoblox::Grid::DNS::Nsgroup->new(
    name        => "ns_group_A",
    primary     => $member1,
    secondaries => [$member2, $nameserver1, $nameserver2,$nameserver3]
    );

my @result_array = $session->get( "object" => "Infoblox::Grid::DNS" ,  "grid" => "Infoblox" );

 my $dns = $result_array[0];
 $dns->nsgroups([$nsg1]);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an Nsgroup object

name( )

Use this method to set or retrieve the name server group, providing a meaningful reference for the set of servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the nameserver group in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting name
 my $name = $nsg1->name();
 #Modifying name
 $nsg1->name("ns_group_2");
```

comment( )

Use this method to set or retrieve the comment field of the Nsgroup object. You can enter additional information about the NS group in this field.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Comment of the nameserver group in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting comment
my $comment = $nsg1->comment();
#Modifying comment
$nsg1->comment("Some comment");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with an Nsgroup object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $nsg1->extensible_attributes();
#Modify extensible attributes
$nsg1->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

primary( )

Use this method to set or retrieve the grid primary server or external primary servers of the nameserver group.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an Infoblox::DNS::Member object for the grid primary server or an array reference that contains Infoblox::DNS::Nameserver and Infoblox::DNS::MSServer objects for external primary servers.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#For grid primary:

my $member1 = Infoblox::DNS::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
    stealth  => "true"
);

$nsg1->primary($member1);

#For external primary:

my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name     => "ns1.domain.extra",
    ipv4addr => "5.5.5.5",
    stealth  => "false",
    TSIGname => "tsig_name",
    TSIGkey  => "ZX1jVJl7C58GT/sc7Q3ucA=="
);

my $nameserver2 = Infoblox::DNS::Nameserver->new(
    name     => "ns2.domain.extra",
    ipv4addr => "6.6.6.6",
    stealth  => "false",
    TSIGname => "tsig_name2",
```

```
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
);

#Creating a nameserver object.
my $nameserver3 = Infoblox::DNS::Nameserver->new(
    name     => "ns3.test.extra",
    ipv6addr => "12ab::34a",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
    );

$nsg1->primary([$nameserver1, $nameserver2,$nameserver3]);
```

ms_ad_integrated()

Use this method to set or retrieve the Active Directory-integrated flag of the nameserver group.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the Active Directory-integrated flag or "false" to unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ms_ad_integrated
my $ms_ad_integrated = $nsgroup->ms_ad_integrated();
#Modify ms_ad_integrated
$nsgroup->ms_ad_integrated("true");
```

secondaries( )

Use this method to set or retrieve the grid secondary server or external secondary servers of the nameserver group.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DNS::Member objects for grid secondary servers, Infoblox::DNS::Nameserver objects for external secondary servers, and Infoblox::DNS::MSServer for Microsoft DNS servers.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
my $member2 = Infoblox::DNS::Member->new(
    name     => "member2.domain.com",
    ipv4addr => "2.2.2.2",
    stealth  => "false"
);

my $nameserver1 = Infoblox::DNS::Nameserver->new(
    name     => "ns1.domain.extra",
    ipv4addr => "4.4.4.4",
    stealth  => "false",
    TSIGname => "tsig_name",
    TSIGkey  => "ZX1jVJl7C58GT/sc7Q3ucA=="
);

my $nameserver2 = Infoblox::DNS::Nameserver->new(
    name     => "ns2.domain.extra",
    ipv4addr => "5.5.5.5",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
);

#Creating a nameserver object.
```

```
my $nameserver3 = Infoblox::DNS::Nameserver->new(
    name     => "ns3.test.extra",
    ipv6addr => "12ab::34a",
    stealth  => "false",
    TSIGname => "tsig_name2",
    TSIGkey  => "qwqwqwqwqwqwqwqwqwqwqA=="
    );

$nsg1->secondaries([$member2, $nameserver1, $nameserver2,$nameserver3]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as modify and remove. This sample also includes error handling for the operations.

#Preparation prior to an Nsgroup object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create an Nsgroup object

```
#Creating a member object.
my $member1 = Infoblox::DNS::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
    stealth  => "false"
);

#Creating a nsgroup object.
my $nsg1 = Infoblox::Grid::DNS::Nsgroup->new(
    name    => "ns_group_1",
    primary => $member1
);

my $nsg1_A = Infoblox::Grid::DNS::Nsgroup->new(
    name    => "ns_group_A",
    primary => $member1
);

unless ($nsg1) {
   die("Construct Nsgroup failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Nsgroup object created successfully\n";
```

#Get and add the Nsgroup object

```
my @result_array = $session->get( "object" => "Infoblox::Grid::DNS" ,  "grid" => "Infoblox" );

if( defined @result_array  and scalar( @result_array ) > 0 )
{
       my $object = $result_array[0];
       if( ref( $object ) eq "Infoblox::Grid::DNS" )
       {

               $object->nsgroups( [ $nsg1 ,$nsg1_A] );

               my $response = $session->modify( $object );
               print "Add Nsgroup to grid dns properties success.\n" if ( $response );
                       }

 unless ($object) {
```

```
    die("Add Nsgroup to grid dns properties failed: ",
        $session->status_code() . ":" . $session->status_detail());
 }
  }

#Modify the Nsgroup object

 my @result_array = $session->get( "object" => "Infoblox::Grid::DNS" ,  "grid" => "Infoblox" );

 if( defined @result_array  and scalar( @result_array ) > 0 )
 {
       my $object = $result_array[0];
       if( ref( $object ) eq "Infoblox::Grid::DNS" )
       {

          $nsg1_A->name("ns_group_modified");

         $object->nsgroups( [ $nsg1 ,$nsg1_A] );

               my $response = $session->modify( $object );
               print "Modify Nsgroup success.\n" if ( $response );
                       }

  unless ($object) {
    die("Modify NSgroup failed: ",
        $session->status_code() . ":" . $session->status_detail());
 }
  }

#Remove a Nsgroup object

 my @result_array = $session->get( "object" => "Infoblox::Grid::DNS" ,  "grid" => "Infoblox" );

 if( defined @result_array  and scalar( @result_array ) > 0 )
 {
       my $object = $result_array[0];
       if( ref( $object ) eq "Infoblox::Grid::DNS" )
       {

               $object->nsgroups( [ ] );

               my $response = $session->modify( $object );
               print "Remove Nsgroup success.\n" if ( $response );
                       }

  unless ($object) {
    die("Remove Nsgroup failed: ",
        $session->status_code() . ":" . $session->status_detail());
 }
  }

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(),Infoblox::Session->modify(),Infoblox::Grid::DNS, Infoblox::DNS::Member,Infoblox::DNS::Nameserver, Infoblox::DNS::MSServer

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::ExtensibleAttributeDef - Extensible Attribute Definition object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - allowed_object_types( )
  - audit_log( )
  - comment( )
  - default_value( )
  - list_values( )
  - max( )
  - min( )
  - multiple( )
  - name( )
  - required( )
  - type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::ExtensibleAttributeDef - Extensible Attribute Definition object.

DESCRIPTION

The Extensible Attribute Definition object is used to create or modify the definition of an extensible attribute. After the definition is created, the attribute can be defined on other Infoblox objects. DHCP Fixed Address, DHCP Fixed Address Template, DHCP Network, DHCP Network Template, DHCP Range, DHCP Range Template, DNS Host, DHCP FailOver and DNS Zone objects support extensible attributes.

CONSTRUCTOR

```
 my $ext_attr_def = Infoblox::Grid::ExtensibleAttributeDef->new(
     name                => $string,                                                    #Required
     type                => "string" | "integer" | "email" | "date" | "list" | "url",     #Optional
/ Default is "string"
     comment             => $string,                                                    #Optional /
Default is undefined
     audit_log           => "true" | "false"                                            #Optional
/ Default is "false"
     required            => "true" | "false"                                            #Optional
/ Default is "false"
     multiple            => "true" | "false"                                            #Optional
/ Default is "false"
     default_value       => $string | $num,                                             #Optional
/ Default is undefined
     list_values         => [$Value1, $Value2, ...],                                    #Optional
/ Default is undefined
     min                 => $num,                                                       #Optional /
Default is undefined
     max                 => $num,                                                       #Optional /
Default is undefined
     allowed_object_types=> [$string, $string, ...]                                     #Optional
/ Default is undefined
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to an Extensible Attribute Definition object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
# Construct an Extensible Attribute Definition object
my $ext_attr_def = Infoblox::Grid::ExtensibleAttributeDef->new(
    name                => "test_ext_attr",
    comment             => "test extensible attribute",
    type                => "integer"
);
# Submit for addition
my $response = $session->add( $ext_attr_def );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific Extensible Attribute Definition object:

 name      - Required. Extensible Attribute Definition name in string format.
```

Example

```
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::ExtensibleAttributeDef",
    name       => "test_ext_attr" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment.
$ext_attr_def->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $ext_attr_def );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the Extensible Attribute Definition object matching the given name
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::ExtensibleAttributeDef",
    name       => "test_ext_attr" );
# Find the desired object from the retrieved list.
my $ext_attr_def = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $ext_attr_def );
```

Infoblox::Session->search( )

Use this method to search for Extensible Attribute Definition objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific Extensible Attribute Definition object:

 name      - Optional. Extensible Attribute Definition name (regular expression).
 comment   - Optional. Extensible Attribute Definition comment (regular expression).
```

Example

```
# search for all Extensible Attribute Definition objects starting with "test"
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::ExtensibleAttributeDef",
    name   => "test.*" );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of an Extensible Attribute Definition object.

allowed_object_types( )

Use this method to set or retrieve the object types on which the extensible attribute can be defined.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Parameter is an array reference that contains the list of object types on which the extensible attribute can be defined. Currenty the supported object types are Infoblox::DHCP::FixedAddr, Infoblox::DHCP::FixedAddrTemplate, Infoblox::DHCP::Network, Infoblox::DHCP::NetworkTemplate, Infoblox::DHCP::IPv6Network, Infoblox::DHCP::SharedNetwork, Infoblox::DHCP::Range, Infoblox::DHCP::RangeTemplate, Infoblox::DNS::Host, Infoblox::DNS::Zone and Infoblox::IFMap::Client. If the parameter is undefined the extensible attribute can be defined on all object types. The default value for this field is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get allowed object types
my $ref_allowed_object_types = $ext_attr_def->allowed_object_types();
# Modify allowed object types
$ext_attr_def->allowed_object_types(['Infoblox::DHCP::Range', 'Infoblox::DHCP::RangeTemplate']);
```

audit_log( )

Use this method to set or retrieve the "audit_log" flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the audit_log flag or "false" to deactivate/unset it. When the flag is set the extensible attribute name and value assigned to an object are logged in the audit log when a change occurs on the object. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get audit_log flag
my $audit_log = $ext_attr_def->audit_log();
# Modify audit_log flag
$ext_attr_def->audit_log("true");
```

comment( )

Use this method to set or retrieve the comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Comment in string format with a maximum of 256 characters. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $ext_attr_def->comment();
#Modify comment
```

```
$ext_attr_def->comment("This is the modified comment for extensible attribute definition");
```

default_value( )

Use this method to set or retrieve the value used to pre-populate the attribute value in the GUI.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Default value used to pre-populate the attribute value in the GUI. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Note that, for a date, the time value is automatically modified to 11:01:00 before insertion, so you may not get the same value that you've inserted upon retrieval. e.g. 1000000 (1970-01-12T13:46:40Z) will be returned as 990060 (1970-01-12T11:01:00Z).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get default value of an extensible attribute of type string
my $ext_attr_def = $ext_attr_def->default_value();
# Modify default value
$ext_attr_def->default_value("Router");
```

list_values( )

Use this method to set or retrieve the list of values that can be defined for the list type.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Parameter is an array reference that contains Infoblox::Grid::ExtensibleAttributeDef::ListValue values associated with an extensible attribute object defined as the "list" type. This parameter is not used for other types. The default value for this field is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get list of values
my $ref_list_values = $ext_attr_def->list_values();

# Add a new value to the list
my $ref_list_values = $ext_attr_def->list_values();
push @$ref_list_values, Infoblox::Grid::ExtensibleAttributeDef::ListValue->new(value => 'Santa
Clara');
$ext_attr_def->list_values($ref_list_values);

# Rename a value in the list
my $ref_list_values = $ext_attr_def->list_values();
foreach my $value (@$ref_list_values)
{
   if ($value->value() eq 'San Francisco')
   {
       $value->value('Los Angeles');
   }
}
$ext_attr_def->list_values($ref_list_values);
```

max( )

Use this method to set or retrieve the maximal length of string for string type or maximal value for integer type.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

For a string, the value is an integer from 1 through 256. For an integer, the value is an integer from -2147483648 through 2147483647. This parameter is not used for other types. Using the Infoblox::Session->modify() function you can only define a value greater than the current one. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get max
my $ext_attr_def = $ext_attr_def->max();
# Modify max
$ext_attr_def->max(23);
```

min( )

Use this method to set or retrieve the minimal length of string for string type or minimal value for integer type.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

For a string, the value is an integer from 1 through 256. For an integer, the value is an integer from -2147483648 through 2147483647. This parameter is not used for other types. Using the Infoblox::Session->modify() function you can only define a value lesser than the current one. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get min
my $ext_attr_def = $ext_attr_def->min();
# Modify min
$ext_attr_def->min(3);
```

multiple( )

Use this method to set or retrieve the "multiple" flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the multiple flag or "false" to deactivate/unset it. When the flag is set multiple values can be assigned to an extensible attribute. Using the Infoblox::Session->modify() function you can change this flag from "false" to "true". The reverse operation is not allowed. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get multiple flag
my $audit_log = $ext_attr_def->multiple();
# Modify multiple flag
$ext_attr_def->multiple("true");
```

name( )

Use this method to set or retrieve the name of an extensible attribute definition.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Name of the extensible attribute with a maximum of 128 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $name = $ext_attr_def->name();
# Modify name
$ext_attr_def->name("Class");
```

required( )

Use this method to set or retrieve the "required" flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the required flag or "false" to deactivate/unset it. When the flag is set the assignment of a value for this attribute is required when adding or modifying a single object in the GUI. The default value for this field is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get required flag
my $required = $ext_attr_def->required();
# Modify required flag
$ext_attr_def->required("false");
```

type( )

Use this method to set or retrieve the type of an extensible attribute.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Parameter is a string. The default value for this field is "string". Currenty the supported object types are 'string', 'integer', 'email', 'date', 'list' and 'url'. Modification of the type using the Infoblox::Session->modify() function is not allowed.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get type
my $type = $ext_attr_def->type();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, get, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to an Extensible Attribute Definition object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
```

```
        die("Construct session failed: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create an Extensible Attribute Definition object

```
my $ext_attr_def = Infoblox::Grid::ExtensibleAttributeDef->new(
    name                => "test_extensible_atttribute",
    comment             => "test extensibleatttribute",
    type                => "integer",
    min                 => -50,
    max                 => 50,
);

unless ($ext_attr_def) {
        die("Construct extensible attribute definition object failed: ",
                Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Extensible Attribute Definition object created successfully\n";

#Add this object to the Infoblox appliance through a session
$session->add($ext_attr_def)
        or die("Add Extensible Attribute Definition object failed: ",
                        $session->status_code() . ":" . $session->status_detail());
print "Extensible Attribute Definition object added to server successfully\n";
```

#Get and modify an Extensible Attribute Definition object

```
#Get the object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::ExtensibleAttributeDef",
    name   => "test_extensible_atttribute"
);
my $object = $retrieved_objs[0];

unless ($object) {
        die("Get Extensible Attribute Definition object failed: ",
                $session->status_code() . ":" . $session->status_detail());
}
print "Get Extensible Attribute Definition object found at least 1 matching entry\n";

#Modify the object
$object->comment("this is a new comment");
$object->min(-100);
$object->max(100);

#Apply the changes.
$session->modify($object)
        or die("Modify Extensible Attribute definition object failed: ",
                $session->status_code() . ":" . $session->status_detail());
print "Extensible Attribute definition object modified successfully \n";

#Accessor methods
print "name: " . $object->name() . "\n";
print "comment: " . $object->comment() . "\n";
print "min: " . $object->min() . "\n";
print "max: " . $object->max() . "\n";
```

#Remove an Extensible Attribute Definition object

```
#Get the object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::ExtensibleAttributeDef",
    name   => "test_extensible_atttribute"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Extensible Attribute Definition object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Extensible Attribute Definition object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
        or die("Remove Extensible Attribute Definition object failed: ",
                $session->status_code() . ":" . $session->status_detail());
```

```
print "Extensible Attribute Definition object removed successfully \n";
####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::ExtensibleAttributeDef::ListValue - Extensible Attribute Definition List Value object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- METHODS
  - value( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::ExtensibleAttributeDef::ListValue - Extensible Attribute Definition List Value object.

DESCRIPTION

The Extensible Attribute Definition List Value object is used to store values for an extensible attribute object defined as the "list" type.

CONSTRUCTOR

```
my $ext_attr_def = Infoblox::Grid::ExtensibleAttributeDef::ListValue->new(
    value                => $string,                                          #Required
);
```

METHODS

This section describes all the methods that you can use to set and retrieve the values of an Extensible Attribute Definition List Value object.

value( )

Use this method to set or retrieve the value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Parameter is a string.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get value
my $value = $ext_attr_def_value->value();
#Modify value
$ext_attr_def_value->value($value);
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::ExtensibleAttributeDef

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::FileDistributionDir - Grid FileDistributionDir object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - name( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::FileDistributionDir - Grid FileDistributionDir object.

DESCRIPTION

The FileDistributionDir object is used to add, get, modify, and remove directories that store files for a file distribution service.

CONSTRUCTOR

```
my $dir = Infoblox::Grid::FileDistributionDir->new(
    name        => $string,            #Required
);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a FileDistributionDir object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the file distribution directory object
my $dir = Infoblox::Grid::FileDistributionDir->new(
    name        => "mydir",
);
# Submit for addition
my $response = $session->add( $dir );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific FileDistributionDir object:

 name          - Required. User name in string format.
```

Example

```
my @retrieved_objs = $session->get(
    object      => "Infoblox::Grid::FileDistributionDir",
    name        => "mydir" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to rename a file distribution directory.
$dir->name("mynewdir");
# Submit modification
my $response = $session->modify( $dir );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the user objects with the same starting name
my @retrieved_objs = $session->get(
    object      => "Infoblox::Grid::FileDistributionDir",
    name        => "mydir" );
# find the desired object from the retrieved list.
my $dir = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $dir );
```

Infoblox::Session->search( )

Use this method to search for FileDistributionDir objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific FileDistributionDir object:

  name          - Required. Directory name in string format. Only the exact match is supported (that
is, exactly the same as 'get' method).
```

Example

```
# search for all FileDistributionDir objects with name "mydir"
my @retrieved_objs = $session->search(
    object      => "Infoblox::Grid::FileDistributionDir",
    name        => "mydir");
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a FileDistributionDir object.

name( )

Use this method to set or retrieve the name of a file distribution directory.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name of the file distribution directory. Enter the complete path to the file distribution directory, including all existing intermediate directories.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name of the file distribution directory
my $name = $file_distribution->name();
#Modify name of the file distribution directory (i.e. rename the directory)
$file_distribution->name("mynewdir");
#Create a new subdirectory 'subdir' under 'mynewdir' ('mynewdir' must exist)
$file_distribution->name("mynewdir/subdir");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and

remove. This sample also includes error handling for the operations.

#Preparation prior to a FileDistributionDir object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
               );

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a FileDistributionDir object

```
my $dir = Infoblox::Grid::FileDistributionDir->new(
    name        => "mydir",
);

unless ($dir) {
       die("Construct file distribution directory object failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "FileDistributionDir object created successfully\n";

#Add the file distribution dir object to the Infoblox appliance through a session
$session->add($dir)
       or die("Add FileDistributionDir object failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "FileDistributionDir object added to server successfully\n";
```

#Search for a file distribution directory

```
my @retrieved_objs = $session->search(
               object      => "Infoblox::Grid::FileDistributionDir",
               name        => "mydir"
              );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search FileDistributionDir object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Search FileDistributionDir object found at least 1 matching entry\n";
```

#Get and modify a FileDistributionDir object

```
#Get the file distribution dir object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
            object      => "Infoblox::Grid::FileDistributionDir",
            name        => "mydir",
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get FileDistributionDir object failed: ",
               $session->status_code() . ":" . $session->status_detail());
}
print "Get FileDistributionDir object found at least 1 matching entry\n";

#Modify the name of the FileDistributionDir object (that is, rename the directory)
$object->name("mynewdir");

#Apply the changes.
$session->modify($object)
       or die("Modify User object failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "FileDistributionDir object modified successfully \n";
```

#Remove a FileDistributionDir object

```
#Get a FileDistributionDir object through the session
my @retrieved_objs = $session->get(
            object      => "Infoblox::Grid::FileDistributionDir",
            name        => "mynewdir",
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get FileDistributionDir object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get FileDistributionDir object found at least 1 matching entry\n";

#Submit an object for removal
$session->remove($object)
       or die("Remove FileDistributionDir object failed: ",
               $session->status_code() . ":" . $session->status_detail());
print "FileDistributionDir object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member::FileDistribution, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::GlobalSmartFolder - Global Smart Folder object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - enable_no_values( )
  - group_bys( )
  - name( )
  - query_items( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::GlobalSmartFolder - Global Smart Folder object.

DESCRIPTION

A Global Smart Folder object is used for data organization.

CONSTRUCTOR

```
 my $group = Infoblox::Grid::GlobalSmartFolder->new(
     name                => $string,                                                    #Required
     comment             => $string,                                                    #Optional /
Default is undefined
     group_bys           => [$GroupbyItem1, $GroupbyItem2],                              #Optional
/ Default is undefined
     query_items         => [$QueryItem1, $QueryItem2],                                  #Optional
/ Default is undefined
     enable_no_values    => "true" | "false",                                            #Optional
/ Default is "false"
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a Global Smart Folder object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the QueryItem in-memory object
my $query_item1 = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                => "location",
    operator            => "eq",
    value               => "Santa Clara",
);

#Construct the GlobalSmartFolder object
my $group_by_state = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                        enable_grouping => 'true',
                                                        value => "State",
                                                        value_type => "EXTATTR",
                                                       );

my $smart_folder = Infoblox::Grid::GlobalSmartFolder->new(
    name                => "smartfolder1",
```

```
    comment              => "global smart folder",
    group_bys            => [$group_by_state],
    query_items          => [$query_item1],
);
# Submit for addition
my $response = $session->add( $smart_folder );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific GlobalSmartFolder object:

 name      - Required. GlobalSmartFolder name in string format.
 comment   - Optional. Comment in string format.
```

Example

```
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::GlobalSmartFolder",
    name       => "smartfolder1" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment.
$smart_folder->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $smart_folder );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the GlobalSmartFolder objects using a name
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::GlobalSmartFolder",
    name       => "smartfolder1" );
# find the desired object from the retrieved list.
my $desired_smart_folder = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_smart_folder );
```

Infoblox::Session->search( )

Use this method to search for Global Smart Folder objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific GlobalSmartFolder object:

 name      - Optional. GlobalSmartFolder name in string format (regular expression).
 comment   - Optional. Comment in string format (regular expression).
```

Example

```
# search for all GlobalSmartFolder objects that start with "smartfolder"
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::GlobalSmartFolder",
    name   => "smartfolder.*" );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a GlobalSmartFolder object.

comment( )

Use this method to set or retrieve the comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Pertinent information about the global smart folder. Comment in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $smart_folder->comment();
#Modify comment
$smart_folder->comment("This is the modified comment for smartfolder1");
```

enable_no_values( )

Use this method to set or retrieve the enable_no_values flag. This flag specifies whether objects that do not contain values for the specified group_by attributes are included in the group for the global smart folder. This flag is disabled by default. To include such objects, enable the option by setting enable_no_values to "true".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_no_values
my $enable_no_values = $smart_folder->enable_no_values();
#Modify enable_no_values
$smart_folder->enable_no_values("true");
```

group_bys( )

Use this method to set or retrieve the group_bys value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::Grid::SmartFolder::GroupBy object(s). The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $group_by_state = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                            enable_grouping => 'true',
                                                            value => "State",
                                                            value_type => "EXTATTR",
                                                           );
#Get group_by value
my $group_by = $smart_folder->group_bys();
#Modify group_bys value
$smart_folder->group_bys([$group_by_state]);
```

name( )

Use this method to set or retrieve the name of a global smart folder.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name of the global smart folder.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $smart_folder->name();
```

query_items( )

Use this method to set or retrieve the query items. If query items are not specified when you `add()` a smart folder, the server automatically assigns a default query item for the smart folder. This default query item is "type=Network/Zone/Range/Member". This ensures that objects are included in searches for these 4 types, by default. When you `modify()` a smart folder, the server keeps the query items provided by the client and does not add the default items.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::Grid::SmartFolder::QueryItem object(s). The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get query items
my $query_items = $smart_folder->query_items();

#Modify query items
my $query_item1 = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                => "location",
    operator            => "eq",
    value               => "Santa Clara",
);
$smart_folder->query_items([$query_items1]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to an GlobalSmartFolder object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

```
# Add a network with extensible attributes

my %extensible_attributes=(
    'Region' => 'Santa Clara County',
    'Country' => 'USA',
    'State' => 'California',
    'Site' => 'Santa Clara',
    'Building' => 'HQ',
    'VLAN' => '1',
    );

my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/24",
    extensible_attributes => \%extensible_attributes,
    );
unless ($network) {
  die("Construct Network failed: ",
      Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network created successfully\n";

$session->add($network)
  or die("Add Network object failed: ",
      $session->status_code() . ":" . $session->status_detail());
print "Network object added to server successfully\n";

B<#Create a GlobalSmartFolder object>

my $query_item = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                    => "Country",
    is_extensible_attribute  => "true",
    operator                => "eq",
    value                   => "USA",
    );

unless ($query_item) {
  die("Construct query item object failed: ",
      Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Smart folder query item object created successfully\n";

my $group_by_country = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                            enable_grouping => 'true',
                                                            value => "Country",
                                                            value_type => "EXTATTR",
                                                           );

my $smart_folder = Infoblox::Grid::GlobalSmartFolder->new(
    name                    => "my_folder_global",
    comment                 => "my global folder",
    query_items              => [$query_item],
    group_bys               => [$group_by_country],
    );

unless ($smart_folder) {
  die("Construct global smart folder object failed: ",
      Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Smart folder object created successfully\n";

#Add the GlobalSmartFolder object to the Infoblox appliance through a session
$session->add($smart_folder)
  or die("Add GlobalSmartFolder object failed: ",
      $session->status_code() . ":" . $session->status_detail());
print "GlobalSmartFolder object added to server successfully\n";

B<#Search for a GlobalSmartFolder>

my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::GlobalSmartFolder",
    name   => "my_folder.*"
    );
my $object = $retrieved_objs[0];

unless ($object) {
  die("Search for a GlobalSmartFolder object failed: ",
      $session->status_code() . ":" . $session->status_detail());
```

```
}
print "Search for a GlobalSmartFolder object found at least 1 matching entry\n";

B<#Get and modify a GlobalSmartFolder object>

#Get the GlobalSmartFolder object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::GlobalSmartFolder",
    name   => "my_folder_global"
    );
my $object = $retrieved_objs[0];

unless ($object) {
  die("Get GlobalSmartFolder object failed: ",
      $session->status_code() . ":" . $session->status_detail());
}
print "Get GlobalSmartFolder object found at least 1 matching entry\n";

#Modify the GlobalSmartFolder object
$object->name("my_folder_global_renamed");
$object->comment("modified global smart folder");

#Apply the changes.
$session->modify($object)
  or die("Modify GlobalSmartFolder object failed: ",
      $session->status_code() . ":" . $session->status_detail());
print "GlobalSmartFolder object modified successfully \n";

B<#Remove a GlobalSmartFolder object>

#Get the GlobalSmartFolder object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::GlobalSmartFolder",
    name   => "my_folder_global_renamed"
    );
my $object = $retrieved_objs[0];
unless ($object) {
  die("Get GlobalSmartFolder object failed: ",
      $session->status_code() . ":" . $session->status_detail());
}
print "Get GlobalSmartFolder object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
  or die("Remove GlobalSmartFolder object failed: ",
      $session->status_code() . ":" . $session->status_detail());
print "GlobalSmartFolder object removed successfully \n";

B<#Cleanup>

#Get the Network object through the session
my @retrieved_objs = $session->get(
    object  => "Infoblox::DHCP::Network",
    network => "10.0.0.0/24"
    );
my $object = $retrieved_objs[0];
unless ($object) {
  die("Get Network object failed: ",
      $session->status_code() . ":" . $session->status_detail());
}
print "Get Network object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
  or die("Remove Network object failed: ",
      $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::SmartFolderChildren, Infoblox::Grid::PersonalSmartFolder, Infoblox::Session->add(), Infoblox::Session->get(),

Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::IFMap - interface to configure the IF-MAP server object properties.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - authorization_rules( )
  - compatibility_mode( )
  - enable_ifmap( )
  - enable_logging( )
  - port( )
  - federation_connectors_count( )
  - federation_licenses_count( )
  - federation_retry_interval( )
  - namespaces( )
  - session_timeout( )
  - subscription_buffer_size( )
  - use_dedicated_cert( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::IFMap - interface to configure the IF-MAP server object properties.

DESCRIPTION

The Infoblox appliance can function as a metadata access point (MAP) and implement the IF-MAP protocol defined by the Trusted Computing Group (TCG, www.trustedcomputinggroup.com). The Infoblox::Grid::IFMap object is used to configure IF-MAP server properties and to specify default values for the authorization rules for clients.

CONSTRUCTOR

The Infoblox::Grid::IFMap object is automatically generated upon the successful installation of the Infoblox appliance. It does not require manual construction.

SESSION METHODS

Infoblox::Session->get( )

Use this method to retrieve the IF-MAP server properties. See Infoblox::Session->get( ) for parameters and return values.

Key Reference

The object is unique. There is no need for additional keys to retrieve it.

Example

```
# Retrieve IF-MAP server properties
my @retrieved_objs = $session->get(
     object => "Infoblox::Grid::IFMap",
     );
```

Infoblox::Session->modify( )

Use this method to update the IF-MAP server properties. This method commits the changes applied to the Infoblox::Grid::IFMap object.

Example

```
# Enable IF-MAP service
$gridifmap->enable_ifmap( "true" );

# Update IF-MAP server properties
my $response = $session->modify( $gridifmap );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an IF-MAP Server object.

authorization_rules( )

Use this method to set or retrieve the value of the authorization_rules attribute. This list will determine the default rules for Infoblox::IFMap::Client objects. This is an ordered list with a "first match" rule and the default is "deny". Authorization rules defined at the client level are evaluated before the server default rules.

Parameter

The valid value is an array reference that contains Infoblox::IFMap::AuthRule objects. The default value is an empty list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get authorization_rules
my $ref_authorization_rules = $gridifmap->authorization_rules( );

# Create authorization rule allowing clients to publish Location on IP addresses
my $rule = Infoblox::IFMap::AuthRule->new(
     identifier => "ip-address",
     metadata => "Location",
     operation => "publish",
     permission => "grant"
     );

# Set authorization_rules
$gridifmap->authorization_rules( [$rule] );
```

compatibility_mode( )

Use this method to set or retrieve the value of the compatibility_mode attribute. The value "1.1-2.0" enables backward compatibility with IF-MAP 1.1 in a mixed environment of IF-MAP 1.1 and 2.0 clients.

Parameter

Allowed values are "none" and "1.1-2.0". The default value is "none".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get compatibility_mode
my $compatibility_mode = $gridifmap->compatibility_mode( );

# Enable backward compatibility
$gridifmap->compatibility_mode( "1.1-2.0" );

# Disable backward compatibility
$gridifmap->compatibility_mode( "none" );
```

enable_ifmap( )

Use this method to set or retrieve the value of the enable_ifmap attribute. This enables or disables the IF-MAP service.

Parameter

Allowed values are "true" and "false". The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_ifmap
my $enable_ifmap = $gridifmap->enable_ifmap( );

# Enable IF-MAP service
$gridifmap->enable_ifmap( "true" );

# Disable IF-MAP service
$gridifmap->enable_ifmap( "false" );
```

enable_logging( )

Use this method to set or retrieve the value of the enable_logging attribute. This enables or disables the logging of XML conversations between the IF-MAP server and its clients.

Parameter

Specify "true" to enable logging or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get enable_logging
my $enable_logging = $gridifmap->enable_logging( );

# Enable IF-MAP server XML logging
$gridifmap->enable_logging( "true" );
```

port( )

Use this method to set or retrieve the value of the port attribute. This configures the external port used by the IF-MAP server when it listens to incoming connections.

Parameter

Any number between 1 and 63999 is allowed. There is no default value. The port is required when the IF-MAP service is enabled.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get port
my $port = $gridifmap->port( );

# Set port to 1234
$gridifmap->port( 1234 );
```

federation_connectors_count( )

Use this method to retrieve the value of the federation_connectors_count attribute. It specifies the number of enabled federation connectors on the system.

Parameter

none

Returns

The method returns the attribute value.

Example

```
# Get federation_connectors_count
my $federation_connectors_count = $gridifmap->federation_connectors_count( );
```

federation_licenses_count( )

Use this method to retrieve the value of the federation_licenses_count attribute. It specifies the number of allowed Federation licenses. The value is 0 if there is no Federation license installed.

Parameter

none

Returns

The method returns the attribute value.

Example

```
# Get federation_licenses_count
my $federation_licenses_count = $gridifmap->federation_licenses_count( );
```

federation_retry_interval( )

Use this method to set or retrieve the value of the federation_retry_interval attribute. This specifies the time, in seconds, between two synchronization attempts, in case of failure.

Parameter

An unsigned integer that represents the retry interval in seconds. The value must be above 5. The default value is 30.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get federation_retry_interval
my $federation_retry_interval = $gridifmap->federation_retry_interval( );

# Change it
$gridifmap->federation_retry_interval( 25);
```

namespaces( )

Use this method to set or retrieve the value of the namespaces attribute. This represents the set of namespaces registered in the server.

Parameter

The valid value is an array reference that contains Infoblox::Grid::IFMap::Namespace objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get namespaces
my $ref_namespaces = $gridifmap->namespaces();

# Add a new namespace
my $ns = Infoblox::Grid::IFMap::Namespace->new(
                 prefix => "ib",
                 uri => "http://www.infoblox.com/ib-metadata.xsd"
                 );
push @$ref_namespaces, $ns

# Set namespaces
$gridifmap->namespaces( $ref_namespaces );
```

session_timeout( )

Use this method to retrieve or set the IF-MAP client session timeout value at the system level. If no argument is passed, the method returns the current attribute value.

Parameter

Session timeout in seconds in integer format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get session timeout
my $timeout = $gridifmap->session_timeout( );

# Set session timeout
$gridifmap->session_timeout( 300 );
```

subscription_buffer_size( )

Use this method to retrieve or set the client connection buffer size at the system level. The IF-MAP server stores subscription updates in a buffer for each client session. If no argument is passed, the method returns the current attribute value.

Parameter

Subscription buffer size in KB in integer format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get subscription buffer size
my $buffer_size = $system_ifmap->subscription_buffer_size( );

# Set subscription buffer size
$system_ifmap->subscription_buffer_size( 250 );
```

use_dedicated_cert( )

Use this method to retrieve or set the value of the use_dedicated_cert attribute. If no argument is passed, the method returns the current attribute value. This method also enables or disables the use of a dedicated certificate for the IF-MAP server. You must import or generate a certificate using the Infoblox::Session->import_data( ) or Infoblox::Session->export_data( ) function before using this method. If the value is "false", the IF-MAP server uses the appliance certificate.

Parameter

The valid values are "true" and "false". The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get the use_dedicated_cert attribute
my $use_dedicated_cert = $gridifmap->use_dedicated_cert();

# Enable the dedicated certificate for IF-MAP server
$gridifmap->use_dedicated_cert( "true" );
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as get and modify. This sample code also includes error handling for the operations.

#Preparations

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the appliance.
my $session = Infoblox::Session->new(
    master   => "192.168.1.2", #appliance host ip
    username => "admin",       #appliance user login
    password => "infoblox"     #appliance password
    );
unless( $session ){
    die("Constructor for session failed:",
    Infoblox::status code            Infoblox::status detail
```

```
                        ( ) . ":" .                        ( ));
}
print" Session object created successfully \n";

#Create IF-MAP authorization rule objects
my $ifmap_auth1=Infoblox::IFMap::AuthRule->new(
                 operation => 'publish',
                 permission => 'grant',
                 identifier => 'all',
                 metadata => 'Location',
                );

my $ifmap_auth2=Infoblox::IFMap::AuthRule->new(
                 operation => 'search',
                 permission => 'grant',
                 identifier => 'ip-adress',
                 metadata => 'role'
                );
```

#Get the IF-MAP object from the appliance through a session.

```
my @result = $session->get(
    object => "Infoblox::Grid::IFMap",
    );
unless( @result ){
    die("Get grid IF-MAP Failed: ",
    $session->status_code( ) . ":" . $session->status_detail( ));
}
print" Get grid IF-MAP Object found at least 1 matching entry \n";
```

#Modify an IF-MAP Server object

```
my $grid_ifmap = $result[0];
if ($grid_ifmap) {
 #Applying the changes
 $grid_ifmap->enable_ifmap("true");
 $grid_ifmap->port(1979);
 $grid_ifmap->enable_logging("true");
 $grid_ifmap->authorization_rules([$ifmap_auth1, $ifmap_auth2]);
 $grid_ifmap->compatibility_mode("1.1-2.0");
 $grid_ifmap->session_timeout(300);
 $grid_ifmap->subscription_buffer_size(250);
 my $response = $session->modify($grid_ifmap);
 unless( $response ){
    die("Modify grid IF-MAP Failed: ",
    $session->status_code( ) . ":" . $session->status_detail( ));
 }
 print" Modify grid IF-MAP Successful \n";
}
else {
   print "No grid IF-MAP Object found.";
}

# cleanup
   $grid_ifmap->authorization_rules([]);
   $grid_ifmap->enable_ifmap("false");
   $grid_ifmap->enable_logging("false");
   $grid_ifmap->port(undef);
   $grid_ifmap->compatibility_mode("none");
   $grid_ifmap->session_timeout(180);
   $grid_ifmap->subscription_buffer_size(100);
 my $response = $session->modify($grid_ifmap);
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::AuthRule, Infoblox::IFMap::Client, Infoblox::Session->get( ),Infoblox::Session->modify( ), Infoblox::Session->export_data( ), Infoblox::Session->import_data(), Infoblox::Session, Infoblox::Grid::IFMap::Namespace

COPYRIGHT

Copyright (c) 1998-2011 Infoblox Inc.

Infoblox::Grid::IFMap::Namespace - IF-MAP Namespace object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::IFMap->namespaces( )
- METHODS
  - locked( )
  - prefix( )
  - uri( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::IFMap::Namespace - IF-MAP Namespace object

DESCRIPTION

This object represents a namespace used by the IF-MAP server. This is used in the context of subscriptions. A subscription filter has to use namespace prefixes known by the server. This is validated when creating/updating Infoblox::IFMap::Federation::Subscription objects.

CONSTRUCTOR

```
my $member = Infoblox::Grid::IFMap::Namespace->new(
    prefix        => $prefix,          #Required
    uri           => $uri,            #Required
);
```

MODULE METHODS

The following functions are available to apply to an IF-MAP namespace object.

Infoblox::Grid::IFMap->namespaces( )

Use this function to set or retrieve the list of namespaces. See Infoblox::Grid::IFMap for parameters and return values.

Example

```
# Add a new namespace
my $ns = Infoblox::Grid::IFMap::Namespace->new(
                 prefix => "ib",
                 uri => "http://www.infoblox.com/ib-metadata.xsd"
                 );

# Set namespaces
$gridifmap->namespaces( [$ns] );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of an IF-MAP namespace object.

locked( )

Use this method to retrieve the value of the locked attribute. It indicates whether the Infoblox::Grid::IFMap::Namespace object can be modified. It is set by the server as soon as this namespace is used or referenced by any subscription or metadata.

Parameter

none

Returns

The method returns the attribute value.

Example

```
#Get locked
my $locked = $namespace->locked();
```

prefix( )

Use this method to set or retrieve the value of the prefix attribute.

Parameter

A string that represents the attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get prefix
my $prefix = $namespace->prefix( );

# Change it
$namespace->prefix("meta");
```

uri( )

Use this method to set or retrieve the value of the URI attribute.

Parameter

A valid URI, such as http://www.trustedcomputinggroup.org/2010/IFMAP-METADATA/2.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get URI
my $uri = $namespace->uri( );

# Change it
$namespace->uri("http://www.trustedcomputinggroup.org/2010/IFMAP-METADATA/2");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::Federation::Subscription, Infoblox::Grid::IFMap

---

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::Grid::KerberosKey - Grid kerberos key object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DHCP->gss_tsig_key( )
  - Infoblox::Grid::Member::DHCP->gss_tsig_key( )
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - domain( )
  - principal( )
  - version( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::KerberosKey - Grid kerberos key object.

DESCRIPTION

This object is used to manage kerberos keys for GSS-TSIG.

CONSTRUCTOR

```
#Constructor for an Grid KerberosKey object
 my $key = Infoblox::Grid::KerberosKey->new(
    principal => $string,  #Required.
    domain    => $string,  #Optional / Default is undefined
    version   => $num,     #Optional / Default is undefined
  );
```

MODULE METHODS

The following functions are available to be applied to an KerberosKey object.

Infoblox::Grid::DHCP->gss_tsig_key( )

Use this function to specify a Kerberos key at the grid level on the Infoblox appliance. See Infoblox::Grid::DHCP->gss_tsig_key() for parameters and return values.

Example

```
#Create a Kerberos object.
my $key = Infoblox::Grid::KerberosKey->new(
                principal => 'host/appliance.ivan.test@LADYBUG.LOCAL',
                domain => 'JACKIE.LOCAL',
                version => '3',
              );

#Configure kerberos key on the Infoblox grid object
my $response = $grid->gss_tsig_key($key);
```

Infoblox::Grid::Member::DHCP->gss_tsig_key( )

Use this function to specify a Kerberos key at the member level on the Infoblox appliance. See Infoblox::Grid::Member::DHCP->gss_tsig_key() for parameters and return values.

Example

```
#Create a KerberosKey object.
 my $key = Infoblox::Grid::KerberosKey->new(
```

```
                    principal => 'host/appliance.ivan.test@LADYBUG.LOCAL',
                    domain => 'JACKIE.LOCAL',
                    version => '3',
                  );

#Configure KerberosKey on the Infoblox member object
my $response = $member->gss_tsig_key($key);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a Kerberos key object.

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

Apply the following attributes to get a specific Kerberos key object:

principal - Required. The principal of the Kerberos key object you want to retrieve.

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::KerberosKey",
    name   => "my-principal");
```

Infoblox::Session->remove( )

Use this method to remove a Kerberos key object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific Kerberos key object, and then submit this object for removal.

Example

```
# Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::KerberosKey",
    name   => "my-principal");
# Find the desired object from the retrieved list.
my $desired_kerberos_key = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_kerberos_key );
```

Infoblox::Session->search( )

Use this method to search for Kerberos key objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

Apply the following attributes to search for a Kerberos key object:

principal - Required. The principal of the Kerberos key object you want to retrieve (regular expression is supported).

Example

```
# search for all grid member objects
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::KerberosKey",
    principal => ".*");
```

METHODS

This section describes all the methods that you can used to configure and retrieve the attribute value of a KerberosKey object.

domain( )

Use this method to set or retrieve the domain of the KerberosKey object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired domain in string format with a maximum of 1023 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get domain
my $domain = $key->domain();
#Modify domain
$key->domain("my-domain");
```

principal( )

Use this method to set or retrieve the principal string of the KerberosKey object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired principal in string format with a maximum of 1023 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get principal
my $string = $key->principal();
#Modify principal
$key->principal("my-principal");
```

version( )

Use this method to set or retrieve the version number of the KerberosKey object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter an unsigned integer between 0 and 4294967295.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get version
my $version = $key->version();
#Modify version
$key->version("5");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, get, and remove an object. Also this sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
 use strict;
 use Infoblox;

 #Create a session to the Infoblox appliance
 my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
```

```
    password => "infoblox"
  );
  unless ($session) {
    die("Construct session failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
  }
  print "Session created successfully\n";

#import kerberos key object

 my $result = $session->import_data(
     type       => "keytab",
     path       => "/infoblox/tests/datasets/dnsone.keytab",
     service => 'dhcp',
    );

 unless($result) {
        die("import kerberos key object failed: ",
             Infoblox::status_code() . ":" . Infoblox::status_detail());
 }
 print "import kerberos key object successfully\n";

#retrieve kerberos key object

 my $key = $session->get(
    object    => "Infoblox::Grid::Kerberos",
    principal => 'host/appliance.ivan.test@LADYBUG.LOCAL',
  );

  unless ($key) {
        die("Get KerberosKey object failed: ",
             $session->status_code() . ":" . $session->status_detail());
   }
   print "Get KerberosKey object found at least 1 matching entry\n";

 my $grid_dhcp = $session->get(
    object  => 'Infoblox::Grid::DHCP',
    cluster => 'Infoblox',
    );
 unless ($grid_dhcp) {
        die("Get Grid DHCP object failed: ",
             $session->status_code() . ":" . $session->status_detail());
 }

 $grid_dhcp->gss_tsig_key($key);
 $grid_dhcp->kdc_server('test.com');

 #Applying the changes to appliance through session.
 $session->modify($grid_dhcp)
      or die("modify Grid DHCP properties failed:" ,
       $session->status_code(), $session->status_detail());
 print "Grid DHCP properties modified successfully for assigning KerberosKey object\n";

#Removing an KerberosKey object

 # remove the reference to kerberos key
 $grid_dhcp->gss_tsig_key(undef);
 $grid_dhcp->kdc_server(undef);

 #Applying the changes to appliance through session.
 $session->modify($grid_dhcp)
      or die("modify Grid DHCP properties failed:" ,
       $session->status_code(), $session->status_detail());
 print "Grid DHCP properties modified successfully for unassigning KerberosKey object\n";

 $session->remove(object => 'Infoblox::Grid::KerberosKey',
                  principal => 'host/appliance.ivan.test@LADYBUG.LOCAL')
      or die("remove KerberosKey object failed:" ,
       $session->status_code(), $session->status_detail());
 print "remove KerberosKey object successfully\n";

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Grid::DHCP, Infoblox::Member::DHCP, Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member - Grid Member object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - active_position( )
  - additional_ip_list( )
  - allow_syslog_clients( )
  - alternate_resolver( )
  - bgp_as( )
  - comment( )
  - enable_ha( )
  - enable_member_redirect( )
  - enable_ntp( )
  - enable_snmpv3_query( )
  - enable_snmpv3_traps( )
  - enable_syslog_proxy( )
  - external_ntp_servers_enabled( )
  - exclude_grid_master_as_ntp_server( )
  - extensible_attributes( )
  - gateway( )
  - ha_node2_port_duplex( )
  - ha_node2_port_speed( )
  - ha_port_duplex( )
  - ha_port_speed( )
  - hwid( )
  - ipv4addr( )
  - lan2_gateway( )
  - lan2_ipv4addr( )
  - lan2_mask( )
  - lan2_node2_port_duplex( )
  - lan2_port( )
  - lan2_node2_port_speed( )
  - lan2_port_duplex( )
  - lan2_port_speed( )
  - lan2_router_id( )
  - lan_node2_port_duplex( )
  - lan_node2_port_speed( )
  - lan_port_duplex( )
  - lan_port_speed( )
  - lcd_input( )
  - mask( )
  - master_candidate( )
  - mgmt_gateway( )
  - mgmt_lan( )
  - mgmt_mask( )
  - mgmt_node2_port_duplex( )
  - mgmt_node2_port_speed( )
  - mgmt_port( )
  - mgmt_port_duplex( )
  - mgmt_port_speed( )
  - name( )
  - nat_enabled( )
  - nat_group( )
  - nat_ip_address( )
  - nic_failover_enabled( )
  - node1_ha( )
  - node1_hwid( )
  - node1_lan( )
  - node1_mgmt_gateway( )
  - node1_mgmt_mask( )
  - node1_mgmt_lan( )
  - node1_mgmt_port( )
  - node1_nat( )
  - node1_service_status( )

- vpn_mtu( )
- node1_vpn_on_mgmt( )
- node1_hwtype( )
- node2_ha( )
- node2_hwid( )
- node2_lan( )
- node2_mgmt_gateway( )
- node2_mgmt_mask( )
- node2_mgmt_lan( )
- node2_mgmt_port( )
- node2_nat( )
- node2_service_status( )
- node2_vpn_on_mgmt( )
- node2_hwtype( )
- notification_email_addr( )
- ntp_access_list( )
- ntp_authentication_key( )
- ntp_server( )
- ospf_list( )
- override_enable_member_redirect( )
- platform( )
- prefer_resolver( )
- query_comm_string( )
- remote_console_access( )
- router_ID( )
- service_status( )
- snmp_admin( )
- snmpv3_query_users( )
- static_routes( )
- support_access( )
- syslog_proxy_tcp_port( )
- syslog_proxy_udp_port( )
- syslog_size( )
- syslog_server( )
- time_zone( )
- trap_comm_string( )
- trap_receiver( )
- vpn_on_mgmt( )

- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member - Grid Member object.

DESCRIPTION

A grid member is a single Infoblox appliance or an HA pair of Infoblox appliances that belong to a grid.

CONSTRUCTOR

```
my $grid_member = Infoblox::Grid::Member->new(
    gateway                 => $ipv4addr,                                   #Required
    ipv4addr                => $ipv4addr,                                   #Required
    mask                    => $netmask,                                    #Required
    name                    => $fqdn,                                       #Required
    additional_ip_list      => [$Interface1, $Interface2, ...],              #Optional / Default
value is empty
    allow_syslog_clients    => [$ipv4addr | $ipv4_subnet/prefix, ...] | undef,    #Optional / Default
value is undefined
    alternate_resolver      => $Nameserver | $Member | undef,                #Optional / Default
value is undefined
    bgp_as                  => [$AS1, ...],                                 #Optional / Default
value is empty
    comment                 => $string,                                     #Optional / Default
value is empty
    enable_member_redirect  => "true" | "false",                            #Optional / Default
value is "false"
    enable_ha               => "true" | "false",                            #Optional / Default
value is "false"
    enable_ntp              => "true" | "false",                            #Optional / Default
```

```
value is "false"
     enable_syslog_proxy     => "true" | "false",                                  #Optional / Default
value is "false"
     exclude_grid_master_as_ntp_server=> "true" | "false",                         #Optional / Default
value is "false"
     extensible_attributes   => { $string => $string | $num, $string => [ $string | $num, ... ], ... }
#Optional / Default is undefined
     external_ntp_servers_enabled  => "true" | "false",                            #Optional / Default
value is "false"
     ha_node2_port_duplex    => "Half" | "Full" | undef,                            #Optional / Default
value is undefined
     ha_node2_port_speed     => 10 | 100 | 100 | undef,                            #Optional / Default
value is undefined
     ha_port_duplex          => "Half" | "Full" | undef,                            #Optional / Default
value is undefined
     ha_port_speed           => 10 | 100 | 100 | undef,                            #Optional / Default
value is undefined
     lan2_gateway            => $ipv4addr,                                         #Optional / Default
value is undefined
     lan2_ipv4addr           => $ipv4addr,                                         #Optional / Default
value is undefined
     lan2_mask               => $netmask,                                          #Optional / Default
value is undefined
     lan2_node2_port_duplex  => "Half" | "Full" | undef,                            #Optional / Default
value is undefined
     lan2_node2_port_speed   => 10 | 100 | 100 | undef,                            #Optional / Default
value is undefined
     lan2_port_duplex        => "Half" | "Full" | undef,                            #Optional / Default
value is undefined
     lan2_port_speed         => 10 | 100 | 100 | undef,                            #Optional / Default
value is undefined
     lan2_port               => "true" | "false",                                  #Optional / Default
value is "false"
     lan2_router_id          => $num,                                              #Optional / Default
value is undefined
     lan_node2_port_duplex   => "Half" | "Full" | undef,                            #Optional / Default
value is undefined
     lan_node2_port_speed    => 10 | 100 | 100 | undef,                            #Optional / Default
value is undefined
     lan_port_duplex         => "Half" | "Full" | undef,                            #Optional / Default
value is undefined
     lan_port_speed          => 10 | 100 | 100 | undef,                            #Optional / Default
value is undefined
     lcd_input               => "true" | "false" | undef,                          #Optional / Default
value is undefined
     master_candidate        => "true" | "false",                                  #Optional / Default
value is undefined
     mgmt_gateway            => $ipv4addr,                                         #Optional / Default
value is undefined
     mgmt_lan                => $ipv4addr,                                         #Optional / Default
value is undefined
     mgmt_mask               => $netmask,                                          #Optional / Default
value is undefined
     mgmt_node2_port_duplex  => "Half" | "Full" | undef,                            #Optional / Default
value is undefined
     mgmt_node2_port_speed   => 10 | 100 | 100 | undef,                            #Optional / Default
value is undefined
     mgmt_port_duplex        => "Half" | "Full" | undef,                            #Optional / Default
value is undefined
     mgmt_port_speed         => 10 | 100 | 100 | undef,                            #Optional / Default
value is undefined
     mgmt_port               => "true" | "false",                                  #Optional / Default
value is "false"
     nat_enabled             => "true" | "false",                                  #Optional / Default
value is "false"
     nat_group               => $string | undef,                                   #Optional / Default
value is empty
     nat_ip_address          => $ipv4addr | undef,                                 #Optional / Default
value is undefined
     nic_failover_enabled    => "true" | "false",                                  #Optional / Default
value is "false"
     node1_ha                => $ipv4addr,                                         #Optional / Default
value is undefined
     node1_hwtype            => "IB-RSP" | ""                                      #Optional / Default
value is ""
     node1_lan               => $ipv4addr,                                         #Optional / Default
value is undefined
     node1_mgmt_gateway      => $ipv4addr,                                         #Optional / Default
value is undefined
     node1_mgmt_lan          => $ipv4addr,                                         #Optional / Default
value is undefined
     node1_mgmt_mask         => $netmask,                                          #Optional / Default
value is undefined
     node1_mgmt_port         => "true" | "false",                                  #Optional / Default
```

```
value is undefined
     node1_nat                => $ipv4addr | undef,                                  #Optional / Default
value is undefined
     node1_vpn_on_mgmt        => "true" | "false",                                   #Optional / Default
value is undefined
     node2_ha                 => $ipv4addr,                                          #Optional / Default
value is undefined
     node2_hwtype             => ""                                                  #Optional / Default
value is ""
     node2_lan                => $ipv4addr,                                          #Optional / Default
value is undefined
     node2_mgmt_gateway       => $ipv4addr,                                          #Optional / Default
value is undefined
     node2_mgmt_lan           => $ipv4addr,                                          #Optional / Default
value is undefined
     node2_mgmt_mask          => $netmask,                                           #Optional / Default
value is undefined
     node2_mgmt_port          => "true" | "false",                                   #Optional / Default
value is undefined
     node2_nat                => $ipv4addr | undef,                                  #Optional / Default
value is undefined
     node2_vpn_on_mgmt        => "true" | "false",                                   #Optional / Default
value is undefined
     notification_email_addr => $email,                                              #Optional / Default
value is undefined
     ntp_access_list          => [$ntpaccess1, $ntpaccess2, ... ] | undef,           #Optional / Default
value is undefined
     ntp_authentication_key   => [$NTPKey1, $NTPKey2, ...],                          #Optional / Default
value is undefined
     ntp_server               => [$ntp_server1, $ntp_server2] | undef,               #Optional / Default
value is undefined
     ospf_list                => [$OSPF1, $OSPF2, ...],                              #Optional / Default
value is empty
     override_enable_member_redirect  => "true" | "false",                           #Optional / Default
value is "false"
     platform                 => "Infoblox" | "Cisco" | "Riverbed" | "VNIOS"         #Optional / Default
value is "Infoblox"
     prefer_resolver          => $Nameserver | $Member | undef,                      #Optional / Default
value is undefined
     query_comm_string        => $string | undef,                                    #Optional / Default
value is undefined
     remote_console_access    => "true" | "false" | undef,                           #Optional / Default
value is undefined
     router_ID                => $num,                                               #Optional / Default
value is undefined
     snmp_admin               => $Admin | undef,                                     #Optional / Default
value is undefined
     static_routes            => [$Route1, $Route2, ...],                            #Optional / Default
value is empty
     support_access           => "true" | "false" | undef,                           #Optional / Default
value is undefined
     syslog_proxy_tcp_port    => $num,                                               #Optional / Default
value is 514
     syslog_proxy_udp_port    => $num,                                               #Optional / Default
value is 514
     syslog_server            => [$SyslogServer, ...] | undef,                       #Optional / Default
value is undefined
     syslog_size              => $num,                                               #Optional / Default
value is 300
     time_zone                => $TimeZone | undef,                                  #Optional / Default
value is undefined
     trap_comm_string         => $string | undef,                                    #Optional / Default
value is undefined
     trap_receiver            => [$ipv4addr, ...] | undef,                           #Optional / Default
value is undefined
     vpn_mtu                  => $num,                                               #Optional / Default
value is 1450
     vpn_on_mgmt              => "true" | "false"                                    #Optional / Default
value is "false"
 );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a grid member object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
```

```
my $grid_member = Infoblox::Grid::Member->new(
    name                    => "hostname.com",
    ipv4addr                => "3.0.0.2",
    mask                    => "255.0.0.0",
    gateway                 => "3.0.0.1",
    );
#Submit for addition
my $response = $session->add( $grid_member );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific grid member object:

 name - Required. A name in FQDN (Fully Qualified Domain Name) format.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member",
    name   => "hostname.com");
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify comment of grid member.
$grid_member->comment("This is modified object");
# Submit modification
my $response = $session->modify( $grid_member );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member",
    name   => "hostname.com");
#Find the desired object from the retrieved list.
my $desired_gridmember = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_gridmember );
```

Infoblox::Session->search( )

Use this method to search for grid member objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a grid member object:

 name    - Required. A name in FQDN (Fully Qualified Domain Name) format (regular expression).
 comment - Optional. A comment in string format (regular expression).
 extensible_attributes   - Optional. A hash reference containing extensible attributes.
```

Example

```
# search for all grid member objects
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::Member",
    name   => ".*\\.com");
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a grid member.

active_position( )

Use this method to retrieve the active server of a grid member.

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get active server
my $active_server = $grid_member->active_position();
```

additional_ip_list( )

Use this method to set or retrieve the additional_ip_list of a grid member. This list contains additional interface information that can be used on the member level.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::Member::Interface object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get additional ip list
my $additional_ip_list = $grid_member->additional_ip_list();
#Modify additional ip list
my $interface_obj = Infoblox::Grid::Member::Interface->new(
        subnet_mask => "255.255.255.255",
        anycast     => "false",
        comment     => "advanced ip config",
        ipv4addr    => "4.3.2.1",
        interface   => "LOOPBACK",
);
$grid_member->additional_ip_list([$interface_obj]);
```

allow_syslog_clients( )

Use this method to set or retrieve the allow_syslog_clients list of a grid member. This list specifies the clients where the syslog messages are logged. This is required only when enable_syslog_proxy is set to "true".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 addresses or networks. To undefine allow_syslog_clients, you must disable the enable_syslog_proxy flag.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get allow syslog clients
my $allow_syslog_clients = $grid_member->allow_syslog_clients();
#Modify allow syslog clients
$grid_member->allow_syslog_clients(["10.10.10.1", "10.10.2.0/24"]);
```

alternate_resolver( )

Use this method to set or retrieve the alternate_resolver of a grid member. The alternate resolver receives queries from the appliance if it does not receive a response from the preferred name server first.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid values are Infoblox::DNS::Member object or Infoblox::DNS::Nameserver object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get alternate resolver
my $alternate_resolver = $grid_member->alternate_resolver();
#Modify alternate resolver
my $member1 = Infoblox::DNS::Member->new(
    name           => $host_name,
    ipv4addr       => $host_ip,
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);
$grid_member->alternate_resolver($member1);
#Undef alternate resolver
$grid_member->alternate_resolver(undef);
```

bgp_as( )

Use this method to set or retrieve the BGP configuration for anycast, of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::Grid::Member::BGP::AS object(s). Currently, only one Autonomous System (AS) object per grid member is supported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get AS list
my $as = $grid_member->bgp_as();
#Modify AS list
my $as1 = Infoblox::Grid::Member::BGP::AS->new(
    as              => 269,
    keepalive       => 4,
    holddown        => 16,
    neighbors       => [ $neighbor ]
);
$grid_member->bgp_as([$as1]);
```

comment( )

Use this method to set or retrieve a descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $grid_member->comment();
```

```
#Modify comment
$grid_member->comment("Modified grid member");
```

enable_ha( )

Use this method to set or retrieve the HA state of a member. It indicates whether a member is a single appliance or an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_ha flag or "false" to deactivate/unset it. The default value is "false". If set to "false", then all the "node1_xxxx" and "node2_xxxx" attributes but node1_service_status and node1_hwtype are disabled, and "get" does not return those attributes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ha of member
my $type = $grid_member->enable_ha();
#Modify type of member
$grid_member->enable_ha("true");
```

enable_member_redirect( )

Use this method to set or retrieve the flag that controls whether the member will redirect GUI connections to the grid master.

Setting this method to a defined value implicitly sets the override_enable_member_redirect method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_enable_member_redirect attribute to "false".

Note that when enable_member_redirect is set to a defined value and override_denable_member_redirect is set to "false", the last operation takes precedence. Thus the sequence $object->enable_member_redirect("true"); $object->override_enable_member_redirect("false"); will set override_enable_member_redirect to "false", and the sequence $object->override_enable_member_redirect("false"); $object->enable_member_redirect("true"); will result in override_enable_member_redirect="true".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_member_redirect flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_member_redirect of member
my $type = $grid_member->enable_member_redirect();
#Modify type of member
$grid_member->enable_member_redirect("true");
```

enable_ntp( )

Use this method to set or retrieve the enable_ntp flag of a grid member. Network Time Protocol (NTP) is a protocol designed to synchronize the clocks of computers over a network. This method enables/disables NTP service on the grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_ntp flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable ntp flag
my $enable_ntp = $grid_member->enable_ntp();
#Modify enable ntp flag
$grid_member->enable_ntp("true");
```

enable_snmpv3_query( )

Use this method to enable or disable SNMPv3 queries.

Parameter

Specify "true" to enable SNMPv3 queries or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get enable_snmpv3_query
my $enable_snmpv3_query = $gridmember->enable_snmpv3_query();
#Modify enable_snmpv3_query
$gridmember->enable_snmpv3_query("false");
```

enable_snmpv3_traps( )

Use this method to enable or disable SNMPv3 traps.

Parameter

Specify "true" to enable SNMPv3 traps or "false" to disable it. Default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get enable_snmpv3_traps
my $enable_snmpv3_traps = $gridmember->enable_snmpv3_traps();
#Modify enable_snmpv3_traps
$gridmember->enable_snmpv3_traps("true");
```

enable_syslog_proxy( )

Use this method to set or retrieve the enable_syslog_proxy flag of a grid member. When enabled, the member receives syslog messages from specified devices, such as syslog servers and routers, and then forwards these messages to an external syslog server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_syslog_proxy flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable syslog proxy flag
my $enable_syslog_proxy = $grid_member->enable_syslog_proxy();
#Modify enable syslog proxy flag
$grid_member->enable_syslog_proxy("true");
```

external_ntp_servers_enabled( )

Use this method to set or retrieve the external_ntp_servers_enabled flag of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

When this flag is set to "false", the member does not use the configured list of NTP (Network Time Protocol) servers.

When this flag is set to "true", the member uses the configured list of NTP servers. If no NTP servers are configured on the member, it uses the NTP servers configured on the grid.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get external_ntp_servers_enabled flag
my $external_ntp_servers_enabled= $grid_member->external_ntp_servers_enabled();
#Modify external_ntp_servers_enabled flag
$grid_member->external_ntp_servers_enabled("true");
```

exclude_grid_master_as_ntp_server( )

Use this method to set or retrieve the exclude_grid_master_as_ntp_server flag of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

When this flag is set to "true", the member does not use the grid master as an NTP (Network Time Protocol) time source. The default value of this parameter is "false", which means that by default the Grid Master is used as one of the time sources for Grid Member.

Note that setting this flag to "true" may have a detrimental effect on the database integrity of the member.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get exclude_grid_master_as_ntp_server flag
my $exclude_grid_master_as_ntp_server= $grid_member->exclude_grid_master_as_ntp_server();
#Modify exclude_grid_master_as_ntp_server flag
$grid_member->exclude_grid_master_as_ntp_server("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a Grid Member object

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $member->extensible_attributes();
#Modify extensible attributes
$member->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

gateway( )

Use this method to set or retrieve the gateway IP address of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get gateway
my $gateway = $grid_member->gateway();
#Modify gateway
$grid_member->gateway("3.0.0.1");
```

ha_node2_port_duplex( )

Use this method to specify or retrieve the duplex setting of the HA port on node 2 of an HA grid member. Note: Use this method in combination with ha_node2_port_speed. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are "Half" or "Full". The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ha_node2_port_duplex
my $ha_node2_port_duplex = $obj->ha_node2_port_duplex();
#Modify ha_node2_port_duplex
$obj->ha_node2_port_duplex("Half");
```

ha_node2_port_speed( )

Use this method to set or retrieve the connection speed of the HA port on node #2 of an HA grid member. Note: Use this method in combination with ha_node2_port_duplex. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are 10, 100 or 1000. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ha_node2_port_speed
my $ha_node2_port_speed = $obj->ha_node2_port_speed();
#Modify ha_node2_port_speed
$obj->ha_node2_port_speed(10);
```

ha_port_duplex( )

Use this method to specify or retrieve the duplex setting of the HA port on node 1 of an HA grid member. Note: Use this method in combination with ha_port_speed. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are "Half" or "Full". The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ha_port_duplex
my $ha_port_duplex = $obj->ha_port_duplex();
#Modify ha_port_duplex
$obj->ha_port_duplex("Half");
```

ha_port_speed( )

Use this method to set or retrieve the connection speed of the HA port on node 1 of an HA grid member. Note: Use this method in combination with ha_port_duplex. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are 10, 100 or 1000. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ha_port_speed
my $ha_port_speed = $obj->ha_port_speed();
#Modify ha_port_speed
$obj->ha_port_speed(10);
```

hwid( )

Use this method to retrieve the hardware ID of the member appliance.

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get hardware ID
my $hwid = $grid_member->hwid();
```

ipv4addr( )

Use this method to set or retrieve the IPv4 address of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $grid_member->ipv4addr();
#Modify ipv4addr
$grid_member->ipv4addr("3.0.0.3");
```

lan2_gateway( )

Use this method to set or retrieve the gateway IP address of the LAN2 port interface. Note that to configure the LAN2 port interface, the lan2_port flag must be set to 'true'.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by

decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN2 gateway
my $lan2_gateway = $grid_member->lan2_gateway();
#Modify LAN2 gateway
$grid_member->lan2_gateway("4.0.0.1");
```

lan2_ipv4addr( )

Use this method to set or retrieve the LAN IP address of the LAN2 port interface. Note that to configure the LAN2 port interface, the lan2_port flag must be set to 'true'.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN2 IPv4 address
my $lan2_ipv4addr = $grid_member->lan2_ipv4addr();
#Modify LAN2 IPv4 address
$grid_member->lan2_ipv4addr("4.0.0.2");
```

lan2_mask( )

Use this method to set or retrieve the netmask of the LAN2 port interface. Note that to configure the LAN2 port interface, the lan2_port flag must be set to 'true'.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Netmask address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 255.255.0.0).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN2 mask
my $lan2_mask = $grid_member->lan2_mask();
#Modify LAN2 mask
$grid_member->lan2_mask("255.255.0.0");
```

lan2_node2_port_duplex( )

Use this method to specify or retrieve the duplex setting of the LAN2 port on node 2 of an HA grid member. Note: Use this method in combination with lan2_node2_port_speed. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are "Half" or "Full". The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lan2_node2_port_duplex
my $lan2_node2_port_duplex = $obj->lan2_node2_port_duplex();
#Modify lan2_node2_port_duplex
$obj->lan2_node2_port_duplex("Half");
```

lan2_port( )

Use this method to set or retrieve the LAN2 port flag of a single grid member or node 1 of an HA grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the lan2_port flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN2 port flag
my $lan2_port = $grid_member->lan2_port();
#Modify LAN2 port flag
$grid_member->lan2_port("true");
```

lan2_node2_port_speed( )

Use this method to set or retrieve the connection speed of the LAN2 port on node 2 of an HA grid member. Note: Use this method in combination with lan2_node2_port_duplex. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are 10, 100 or 1000. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lan2_node2_port_speed
my $lan2_node2_port_speed = $obj->lan2_node2_port_speed();
#Modify lan2_node2_port_speed
$obj->lan2_node2_port_speed(10);
```

lan2_port_duplex( )

Use this method to specify or retrieve the duplex setting of the LAN2 port on a single grid member or on node 1 of an HA grid member. Note: Use this method in combination with lan2_port_speed. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are "Half" or "Full". The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lan2_port_duplex
my $lan2_port_duplex = $obj->lan2_port_duplex();
#Modify lan2_port_duplex
$obj->lan2_port_duplex("Half");
```

lan2_port_speed( )

Use this method to set or retrieve the connection speed of the LAN2 port on a single grid member or on node 1 of an HA grid member. Note: Use this method in combination with lan2_port_duplex. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are 10, 100 or 1000. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lan2_port_speed
my $lan2_port_speed = $obj->lan2_port_speed();
#Modify lan2_port_speed
$obj->lan2_port_speed(10);
```

lan2_router_id( )

Use this method to set or retrieve the LAN2 virtual router ID of a grid member. Required only when the member is an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A number with the LAN2 router identification. Valid values are from 1 through 255.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN2 virtual router ID
my $lan2_router_id = $grid_member->lan2_router_id();
#Modify LAN2 virtual router ID
$grid_member->lan2_router_id("123");
```

lan_node2_port_duplex( )

Use this method to specify or retrieve the duplex setting of the LAN port on node 2 of an HA grid member. Note: Use this method in combination with lan_node2_port_speed. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are "Half" or "Full". The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lan_node2_port_duplex
my $lan_node2_port_duplex = $obj->lan_node2_port_duplex();
#Modify lan_node2_port_duplex
$obj->lan_node2_port_duplex("Half");
```

lan_node2_port_speed( )

Use this method to set or retrieve the connection speed of the LAN port on node 2 of an HA grid member. Note: Use this method in combination with lan_node2_port_duplex. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are 10, 100 or 1000. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lan_node2_port_speed
my $lan_node2_port_speed = $obj->lan_node2_port_speed();
#Modify lan_node2_port_speed
$obj->lan_node2_port_speed(10);
```

lan_port_duplex( )

Use this method to specify or retrieve the duplex setting of the LAN port on a single grid member or on node 1 of an HA grid member.. Note: Use this method in combination with lan_port_speed. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are "Half" or "Full". The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lan_port_duplex
my $lan_port_duplex = $obj->lan_port_duplex();
#Modify lan_port_duplex
$obj->lan_port_duplex("Half");
```

lan_port_speed( )

Use this method to set or retrieve the connection speed of the LAN port on a single grid member or on node 1 of an HA grid member.. Note: Use this method in combination with lan_port_duplex. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are 10, 100 or 1000. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lan_port_speed
my $lan_port_speed = $obj->lan_port_speed();
#Modify lan_port_speed
$obj->lan_port_speed(10);
```

lcd_input( )

Use this method to set or retrieve the lcd_input flag of a grid member. This flag enables/disables the Liquid Crystal Display (LCD) input buttons on the front panel of the appliance.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the lcd_input flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lcd input flag
my $lcd_input = $grid_member->lcd_input();
#Modify lcd input flag
$grid_member->lcd_input("true");
#Undef lcd input flag
$grid_member->lcd_input(undef);
```

mask( )

Use this method to set or retrieve the netmask of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Netmask address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 255.255.0.0).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mask
my $mask = $grid_member->mask();
#Modify mask
$grid_member->mask("255.255.0.0");
```

master_candidate( )

Use this method to set or retrieve the master_candidate flag of a grid member. This flag enables the grid member to assume the role of grid master as a disaster recovery measure.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the master_candidate flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get master candidate flag
my $master_candidate = $grid_member->master_candidate();
#Modify master candidate flag
$grid_member->master_candidate("true");
```

mgmt_gateway( )

Use this method to set or retrieve the gateway IP address of the MGMT port.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mgmt gateway
my $mgmt_gateway = $grid_member->mgmt_gateway();
#Modify mgmt gateway
$grid_member->mgmt_gateway("4.0.0.1");
```

mgmt_lan( )

Use this method to set or retrieve the LAN IP address of the MGMT port.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mgmt_lan
my $mgmt_lan = $grid_member->mgmt_lan();
#Modify mgmt_lan
$grid_member->mgmt_lan("4.0.0.2");
```

mgmt_mask( )

Use this method to set or retrieve the netmask of the MGMT port.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Netmask address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 255.255.0.0).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mgmt_mask
my $mgmt_mask = $grid_member->mgmt_mask();
#Modify mgmt_mask
$grid_member->mgmt_mask("255.255.0.0");
```

mgmt_node2_port_duplex( )

Use this method to specify the duplex setting of the MGMT port on node 2 of an HA pair. Note: Use this method in combination with mgmt_node2_port_speed. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are "Half" or "Full". The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mgmt_node2_port_duplex
my $mgmt_node2_port_duplex = $obj->mgmt_node2_port_duplex();
#Modify mgmt_node2_port_duplex
$obj->mgmt_node2_port_duplex("Half");
```

mgmt_node2_port_speed( )

Use this method to set the connection speed of the MGMT port on node 2 of an HA pair. Note: Use this method in combination with mgmt_node2_port_duplex. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are 10, 100 or 1000. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mgmt_node2_port_speed
my $mgmt_node2_port_speed = $obj->mgmt_node2_port_speed();
#Modify mgmt_node2_port_speed
$obj->mgmt_node2_port_speed(10);
```

mgmt_port( )

Use this method to set or retrieve the MGMT port flag of a grid member. You can restrict administrative access to an Infoblox appliance by connecting the MGMT port to a subnet containing only management systems.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Note that node1_mgmt_port, node2_mgmt_port and mgmt_port share the value; setting any of them will have the same effect.

Parameter

Specify "true" to set the mgmt_port flag or "false" to deactivate/unset it. The default value is "false". To disable the mgmt_port flag, you must disable the "node1_mgmt_port" and "node2_mgmt_port" flags as well.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get MGMT port flag
my $mgmt_port = $grid_member->mgmt_port();
#Modify MGMT port flag
$grid_member->mgmt_port("true");
```

mgmt_port_duplex( )

Use this method to specify the duplex setting of the MGMT port on a single grid member or on node 1 of an HA grid member. Note: Use this method in combination with mgmt_port_speed. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are "Half" or "Full". The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mgmt_port_duplex
my $mgmt_port_duplex = $obj->mgmt_port_duplex();
#Modify mgmt_port_duplex
$obj->mgmt_port_duplex("Half");
```

mgmt_port_speed( )

Use this method to specify the connection speed of the MGMT port on a single grid member or on node 1 of an HA grid member. Note: Use this method in combination with mgmt_port_duplex. If any of these attributes are undefined, the default values for duplex and speed settings are used.

Parameter

Valid values are 10, 100 or 1000. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mgmt_port_speed
my $mgmt_port_speed = $obj->mgmt_port_speed();
#Modify mgmt_port_speed
$obj->mgmt_port_speed(10);
```

name( )

Use this method to set or retrieve the host name of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The host name in FQDN (Fully Qualified Domain Name) format of a grid member. The FQDN consists of the host name followed by the domain name (example: abc.com). A host name can have a maximum of 256 bytes.

Note that name matching is not case-sensitive, and that names are inserted in lowercase.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $grid_member->name();
#Modify name
$grid_member->name("hostname2.com");
```

nat_enabled( )

Use this method to set or retrieve the NAT compatibility setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the NAT compatibility setting or "false" to disable it.

This method will also be automatically set to "true" or "false" when "nat_group", "nat_ip_address", "node1_nat" and "node2_nat" change. If any of them is set, nat_enabled will be set to "true"; if all of them are undefined, it will be set to "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nat group
my $nat_group = $grid_member->nat_group();
#Modify nat group
$grid_member->nat_group("nat_group");
```

nat_group( )

Use this method to set or retrieve the NAT group of a grid member. Required only when NAT compatibility is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the NAT (Network Address Translation) group name with a maximum of 256 bytes. Default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nat group
my $nat_group = $grid_member->nat_group();
```

```
#Modify nat group
$grid_member->nat_group("nat_group");
```

nat_ip_address( )

Use this method to set or retrieve the NAT IP address of a grid member. Required only when NAT compatibility is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nat IP address
my $nat_ip_address = $grid_member->nat_ip_address();
#Modify nat ip address
$grid_member->nat_ip_address("5.5.5.5");
```

nic_failover_enabled( )

Use this method to set or retrieve the nic_failover_enabled flag of a grid member. This is required only when the member is a single appliance. To enable this flag, the lan2_port flag must also be set to 'true'.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the nic_failover_enabled flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get NIC failover enabled flag
my $nic_failover_enabled = $grid_member->nic_failover_enabled();
#Modify NIC failover enabled flag
$grid_member->nic_failover_enabled("true");
```

node1_ha( )

Use this method to set or retrieve the HA IP address of node 1. Required only when the member is an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get node1 ha IP address
my $node1_ha = $grid_member->node1_ha();
#Modify node1 ha IP address
$grid_member->node1_ha("3.0.0.4");
```

node1_hwid( )

Use this method to retrieve the hardware ID of node 1 of an HA grid member.

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get node1 hardware ID
my $node1_hwid = $grid_member->node1_hwid();
```

node1_lan( )

Use this method to set or retrieve the LAN IP address of node 1 of an HA grid member. Required only when the member is an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN IP address of node1
my $node1_lan = $grid_member->node1_lan();
#Modify LAN IP address of node1
$grid_member->node1_lan("3.0.0.9");
```

node1_mgmt_gateway( )

Use this method to set or retrieve the gateway IP address of the MGMT port on node 1 of an HA grid member. Required only when the MGMT port is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get gateway of MGMT port of node1
my $node1_mgmt_gateway = $grid_member->node1_mgmt_gateway();
#Modify gateway of MGMT port of node1
$grid_member->node1_mgmt_gateway("6.0.0.1");
```

node1_mgmt_mask( )

Use this method to set or retrieve the netmask of the MGMT port on node 1 of an HA grid member. Required only when the MGMT port is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Netmask address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 255.255.0.0).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get netmask of MGMT port of node 1
my $node1_mgmt_mask = $grid_member->node1_mgmt_mask();
#Modify netmask of MGMT port of node 1
$grid_member->node1_mgmt_mask("255.255.0.0");
```

node1_mgmt_lan( )

Use this method to set or retrieve the LAN IP address of the MGMT port on node 1 of an HA grid member. Required only when the MGMT port is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN IP address of MGMT port on node 1
my $node1_mgmt_lan = $grid_member->node1_mgmt_lan();
#Modify LAN IP address of MGMT port on node 1
$grid_member->node1_mgmt_lan("6.0.0.5");
```

node1_mgmt_port( )

Use this method to set or retrieve the MGMT port flag of an HA grid member. Required only when the MGMT port is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Note that node1_mgmt_port, node2_mgmt_port and mgmt_port share the value; setting any of them will have the same effect.

Parameter

Specify "true" to set the node1_mgmt_port flag or "false" to deactivate/unset it. The default value is undefined. To disable node1_mgmt_port flag we also need to disable "mgmt_port" and "node2_mgmt_port" flags.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get node 1 MGMT port flag
my $node1_mgmt_port = $grid_member->node1_mgmt_port();
#Modify node 1 MGMT port flag
$grid_member->node1_mgmt_port("true");
```

node1_nat( )

Use this method to set or retrieve the Network Address Translation (NAT) address of node 1 of an HA grid member. Required only when NAT compatibility is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get Network Address Translation (NAT) address of node 1
my $node1_nat = $grid_member->node1_nat();
#Modify Network Address Translation (NAT) address of node 1
$grid_member->node1_nat("5.2.2.1");
```

node1_service_status( )

Use this method to retrieve the service status list of a grid member.

Returns

On retrieval, the member returns an array reference that contains Infoblox::Grid::Member::ServiceStatus objects.

Example

```
#Get the service status list
my $services_list = $grid_member->node1_service_status();
```

vpn_mtu( )

Use this method to set or retrieve the VPN maximum transmission unit (MTU).

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the value to set the vpn_mtu value.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get VPN MTU
my $vpn_mtu = $grid_member->vpn_mtu();
#Modify VPN MTU
$grid_member->vpn_mtu(600);
```

node1_vpn_on_mgmt( )

Use this method to set or retrieve the VPN on MGMT port flag of an HA grid member. Valid only when the MGMT port is enabled and the member is not a master candidate. This flag enables VPN services on the MGMT port of an HA grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Note that node1_vpn_on_mgmt, node2_vpn_on_mgmt and vpn_on_mgmt share the value; setting any of them will have the same effect.

Parameter

Specify "true" to set the node1_vpn_on_mgmt flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get node 1 VPN on MGMT port flag
my $node1_vpn_on_mgmt = $grid_member->node1_vpn_on_mgmt();
#Modify node 1 VPN on MGMT port flag
$grid_member->node1_vpn_on_mgmt("false");
```

node1_hwtype( )

Use this method to retrieve the hardware type of node 1 of an HA grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The actual hardware type is populated when the physical node joins the grid. The default value is "".

Returns

The method returns the attribute value.

Example

```
#Get node1 hardware type
my $node1_hwtype = $grid_member->node1_hwtype();
```

node2_ha( )

Use this method to set or retrieve the HA IP address of node 2 of an HA grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get node 2 HA IP address
my $node2_ha = $grid_member->node2_ha();
#Modify node 2 HA IP address
$grid_member->node2_ha("3.0.0.6");
```

node2_hwid( )

Use this method to retrieve the hardware ID of node 2 of an HA grid member.

Parameter

None

Returns

The method returns the attribute value.

Example

```
#Get node2 hardware ID
my $node2_hwid = $grid_member->node2_hwid();
```

node2_lan( )

Use this method to set or retrieve the LAN IP address of node 2 of an HA grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN IP address of node 2
my $node2_lan = $grid_member->node2_lan();
#Modify LAN IP address of node 2
$grid_member->node2_lan("3.0.0.5");
```

node2_mgmt_gateway( )

Use this method to set or retrieve the gateway IP address of the MGMT port on node 2 of an HA grid member. Required only when the MGMT port is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get gateway of MGMT port of node 2
my $node2_mgmt_gateway = $grid_member->node2_mgmt_gateway();
#Modify gateway of MGMT port of node 2
$grid_member->node2_mgmt_gateway("7.0.0.1");
```

node2_mgmt_mask( )

Use this method to set or retrieve the netmask of the MGMT port on node 2 of an HA grid member. Required only when the MGMT port is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Netmask address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 255.255.0.0).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get netmask of MGMT port of node 2
my $node2_mgmt_mask = $grid_member->node2_mgmt_mask();
#Modify netmask of MGMT port of node 2
$grid_member->node2_mgmt_mask("255.255.0.0");
```

node2_mgmt_lan( )

Use this method to set or retrieve the LAN IP address of the MGMT port on node 2 of an HA grid member. Required only when the MGMT port is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get LAN IP address of MGMT port of node 2
my $node2_mgmt_lan = $grid_member->node2_mgmt_lan();
#Modify LAN IP address of MGMT port of node 2
$grid_member->node2_mgmt_lan("7.0.0.5");
```

node2_mgmt_port( )

Use this method to set or retrieve the MGMT port flag of an HA grid member. Required only when the MGMT port is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Note that node1_mgmt_port, node2_mgmt_port and mgmt_port share the value: setting any of them will have the same effect.

Parameter

Specify "true" to set the node2_mgmt_port flag or "false" to deactivate/unset it. The default value is undefined. To disable node2_mgmt_port flag, you must disable the "mgmt_port" and "node1_mgmt_port" flags as well.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get MGMT port flag of node 2
my $node2_mgmt_port = $grid_member->node2_mgmt_port();
#Modify MGMT port flag of node 2
$grid_member->node2_mgmt_port("true");
```

node2_nat( )

Use this method to set or retrieve the Network Address Translation (NAT) address of node 2. Required only when NAT compatibility is enabled.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get Network Address Translation (NAT) address of node2
my $node2_nat = $grid_member->node2_nat();
#Modify Network Address Translation (NAT) address of node2
$grid_member->node2_nat("5.4.4.6");
```

node2_service_status( )

Use this method to retrieve the service status list of a grid member.

Returns

On retrieval, the member returns an array reference that contains Infoblox::Grid::Member::ServiceStatus objects.

Example

```
#Get the service status list
my $services_list = $grid_member->node2_service_status();
```

node2_vpn_on_mgmt( )

Use this method to set or retrieve the VPN on MGMT port flag of an HA grid member. Valid only when the MGMT port is enabled and the member is not a master candidate. This flag enables VPN services on the MGMT port of an HA grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Note that node1_vpn_on_mgmt, node2_vpn_on_mgmt and vpn_on_mgmt share the value: setting any of them will have the same effect.

Parameter

Specify "true" to set the node2_vpn_on_mgmt flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get node 2 VPN on MGMT port flag
```

```
my $node2_vpn_on_mgmt = $grid_member->node2_vpn_on_mgmt();
#Modify node 2 VPN on MGMT port flag
$grid_member->node2_vpn_on_mgmt("false");
```

node2_hwtype( )

Use this method to retrieve the hardware type of node 2 of an HA grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The actual hardware type is populated when the physical node joins the grid. The default value is "".

Returns

The method returns the attribute value.

Example

```
#Get node2 hardware type
my $node2_hwtype = $grid_member->node2_hwtype();
```

notification_email_addr( )

Use this method to set or retrieve the notification email address of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The email address. Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get notification email address
my $notification_email_addr = $grid_member->notification_email_addr();
#Modify notification email address
$grid_member->notification_email_addr("admin\@infoblox.com");
```

ntp_access_list( )

Use this method to set or retrieve the Network Time Protocols (NTP) access control list of a grid member. Valid only when the enable_ntp flag is set to "true".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::NTPAccess objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get NTP access control list
my $ntp_access_list = $grid_member->ntp_access_list();
#Modify NTP access control list
$grid_member->ntp_access_list([$ntpac1,$ntpac2]);
#Undef NTP access control list
$grid_member->ntp_access_list(undef);
```

ntp_authentication_key( )

Use this method to set or retrieve the Network Time Protocol (NTP) authentication key(s) of a grid member. Valid only when the enable_ntp flag is set to "true".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::NTPKey object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get NTP authentication key
my $ntp_authentication_key = $grid_member->ntp_authentication_key();
#Modify NTP authentication key
my $ntp_key1 = Infoblox::Grid::NTPKey->new(
    key_number => "1111",
    key_type   => "M",
    key_string => "aaaa"
);
my $ntp_key2 = Infoblox::Grid::NTPKey->new(
    key_number => "2222",
    key_type   => "n",
    key_string => "d3e54352e5548080"
);
$grid_member->ntp_authentication_key([$ntp_key1, $ntp_key2]);
```

ntp_server( )

Use this method to set or retrieve the NTP server list.

Parameter

Array reference that contains Infoblox::Grid::NTPServer objects or undef. Specifying a reference to an empty array while the "exclude_grid_master_as_ntp_server" attribute is set to "false" sets the Grid Master as the only NTP source for the given Member. Setting the value of this parameter to undef disables the feature.

NOTE: Currently, if you need to specify an external NTP server with authentication, the NTP keys that you use must be known to the Infoblox::Grid object. See the following example.

NOTE: Setting this to a list also sets external_ntp_servers_enabled to 'true'. Setting this to undef also sets external_ntp_servers_enabled to 'false'.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get the ntp_server list for the given member
my $ntp_server_list=$grid_member->ntp_server();

#Modify ntp_server property
my $ntp_server1 = Infoblox::Grid::NTPServer->new(address => "1.1.1.1");
my $ntp_server2 = Infoblox::Grid::NTPServer->new(address => "2.2.2.2");
$grid_member->ntp_server([$ntp_server1,$ntp_server2]);

#Setting grid master as the only NTP source
my $session=Infoblox::Session->new(
        master   => "127.0.0.1",
    username => "admin",
    password => "infoblox"
       );
die(Infoblox::status_code().":".Infoblox::status_detail()) unless $session;

my $grid_member=$session->get(
                              object => "Infoblox::Grid::Member",
                              name   => "infoblox.localdomain"
                             );
die($session->status_code()." : ".$session->status_detail()) unless (ref($grid_member) eq
"Infoblox::Grid::Member");

$grid_member->ntp_server([]);
$grid_member->exclude_grid_master_as_ntp_server("false");
$session->modify($grid_member) or die($session->status_code()." : ".$session->status_detail());

#Adding external NTP servers with authentication
my $session=Infoblox::Session->new(
```

```
        master   => "127.0.0.1",
    username => "admin",
    password => "infoblox"
       );
 die(Infoblox::status_code().":".Infoblox::status_detail()) unless $session;
 my $grid_member=$session->get(
                              object => "Infoblox::Grid::Member",
                              name   => "infoblox.localdomain"
                             );
 die($session->status_code()." : ".$session->status_detail()) unless (ref($grid_member) eq
"Infoblox::Grid::Member");

 my $grid=$session->get(
                        object => "Infoblox::Grid",
                        name   => "Infoblox"
                       );
 die($session->status_code()." : ".$session->status_detail()) unless (ref($grid) eq
"Infoblox::Grid");

 #It is assumed in the following that the grid has NTP enabled
 die("Grid must have ntp enabled) unless ($grid->enable_ntp() eq "true");

 my $ntp_key_1= Infoblox::Grid::NTPKey->new(
           key_number => "101",
           key_type   => "n",
           key_string => "d3e54352e5548080"
       );
 $grid->ntp_authentication_key([$ntp_key_1]);
 $session->modify($grid) or die(($session->status_code()." : ".$session->status_detail());

 my $ntp_server=Infoblox::Grid::NTPServer->new(
                                                address => "1.1.1.1"
                                                authentication => "true"
                                            ntp_key=>$ntp_key_1
                                               );

 $grid_member->ntp_server([$ntp_server]);
 $session->modify($grid_member) or die(($session->status_code()." : ".$session->status_detail());
```

ospf_list( )

Use this method to set or retrieve the OSPF area configuration (for anycast) list of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::Member::OSPF object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
 #Get OSPF list
 my $ospf_list = $grid_member->ospf_list();
 #Modify OSPF list
 my $ospf1 = Infoblox::Grid::Member::OSPF->new(
        area_id            => "1",
        interface          => "LAN_HA",
        authentication_type => "none",
 );

 $grid_member->ospf_list([$ospf1]);
```

override_enable_member_redirect( )

The override_enable_member_redirect attribute controls whether the enable_member_redirect value in the member is used, instead of the grid default.

The override_enable_member_redirect attribute can be specified explicitly. It is also set implicitly when enable_member_redirect is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the override_enable_member_redirect flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_enable_member_redirect of member
my $type = $grid_member->override_enable_member_redirect();
#Modify type of member
$grid_member->override_enable_member_redirect("true");
```

platform( )

Use this method to set or retrieve the platform type of a grid member. If you do not specify a value, the default value is "Infoblox".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is one of either "Infoblox", "Cisco", "Riverbed", or "VNIOS".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get member platform
my $platform = $grid_member->platform();
#Modify node 2 hardware type
$grid_member->platform("Cisco");
```

prefer_resolver( )

Use this method to set or retrieve the preferred name server of the grid member. The grid member sends queries to this name server first.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are Infoblox::DNS::Nameserver object or Infoblox::DNS::Member object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get preferred resolver
my $prefer_resolver = $grid_member->prefer_resolver();
#Modify preferred resolver
my $member1 = Infoblox::DNS::Member->new(
    name           => $host_name,
    ipv4addr       => $host_ip,
    stealth        => "false",
    lead           => "false",
    grid_replicate => "false"
);
$grid_member->prefer_resolver($member1);
#Undef prefer resolver
$grid_member->prefer_resolver(undef);
```

query_comm_string( )

Use this method to set or retrieve the Simple Network Management Protocol (SNMP) query community string that the SNMP management system must send together with its queries to the grid member. An SNMP community string is a text string that acts as a

password. It is used to authenticate messages that are sent between the management station (the SNMP manager) and the grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the Simple Network Management Protocol (SNMP) query community string with a maximum of 50 bytes. Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get query community string
my $query_comm_string = $grid_member->query_comm_string();
#Modify query community string
$grid_member->query_comm_string("test");
#Undef query communication string
$grid_member->query_comm_string(undef);
```

remote_console_access( )

Use this method to set or retrieve the remote_console support access flag of a grid member. When you enable remote console access, a client can access the device using a Secure Shell (SSH) connection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the remote_console_access flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get remote console access flag
my $remote_console_access = $grid_member->remote_console_access();
#Modify remote console access flag
$grid_member->remote_console_access("true");
#Undef remote console access flag
$grid_member->remote_console_access(undef);
```

router_ID( )

Use this method to set or retrieve the virtual router ID of an HA grid member. Required only when the member is an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number with the router identification. Default value is undefined. Valid values should be between 1 and 255

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get router ID
my $router_ID = $grid_member->router_ID();
#Modify router ID
$grid_member->router_ID("34");
```

service_status( )

Use this method to retrieve the service status list of a grid member.

Returns

On retrieval, the member returns an array reference that contains Infoblox::Grid::Member::ServiceStatus objects.

Example

```
#Get the service status list
my $services_list = $grid_member->service_status();
```

snmp_admin( )

Use this method to set or retrieve the Simple Network Management Protocol (SNMP) admin object of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an Infoblox::Grid::SNMP::Admin object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get SNMP admin
my $snmp_admin = $grid_member->snmp_admin();
#Modify SNMP admin
my $snmp_admin_obj = Infoblox::Grid::SNMP::Admin->new(
    sysContact_node1  => "Somebody",
    sysDescr_node1    => "SunOS noc1 5.6 Generic_105181-03 sun4u",
    sysLocation_node1 => "Infoblox Headquarter",
    sysName_node1     => "noc1"
);
$grid_member->snmp_admin($snmp_admin_obj);
#Undef snmp admin
$grid_member->snmp_admin(undef);
```

snmpv3_query_users( )

Use this method to set or retrieve the SNMPv3 user setting at the grid level.

Parameter

Valid value is a reference to the list of the Infoblox::Grid::SNMP::User manpage objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get snmpv3_query_users field
my $snmpv3_users = $gridmember->snmpv3_query_users();
my $user = $$snmpv3_users[0];
#Set snmpv3_query_users field
$gridmember->snmpv3_query_users([$user1, $user2]);
```

static_routes( )

Use this method to set or retrieve the static routes list of a grid member. This list contains the static routes of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::Member::StaticRoute object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get static routes list
my $static_routes = $grid_member->static_routes();
```

```
#Modify static routes list
my $route = Infoblox::Grid::Member::StaticRoute->new(
       "network" => "192.168.2.0/24",
       "gateway" => "192.168.1.1",
);
$grid_member->static_routes([$route]);
```

support_access( )

Use this method to set or retrieve the support access flag of a grid member. This flag enables a Secure Shell (SSH) daemon that only Infoblox Technical Support can access to perform troubleshooting at the grid member level.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the remote_console_access flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get support access flag
my $support_access = $grid_member->support_access();
#Modify support access flag
$grid_member->support_access("true");
#Undef support access
$grid_member->support_access(undef);
```

syslog_proxy_tcp_port( )

Use this method to set or retrieve the Transmission Control Protocol (TCP) listening port for syslog proxy.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

TCP port number. Default value is 514. Valid values should be between 1 and 65535.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get syslog proxy tcp port
my $syslog_proxy_tcp_port = $grid_member->syslog_proxy_tcp_port();
#Modify syslog proxy tcp port
$grid_member->syslog_proxy_tcp_port(514);
```

syslog_proxy_udp_port( )

Use this method to set or retrieve the User Datagram Protocol (UDP) listening port for syslog proxy.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

UDP port number. Default value is 514. Valid values should be between 1 and 65535.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get syslog proxy udp port
my $syslog_proxy_udp_port = $grid_member->syslog_proxy_udp_port();
#Modify syslog proxy udp port
$grid_member->syslog_proxy_udp_port(514);
```

syslog_size( )

Use this method to set or retrieve the grid member syslog rotation size in megabytes.

Parameter

Valid value is between 10 and 300.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get syslog_size
my $syslog_size = $obj->syslog_size();
#Modify syslog_size
$obj->syslog_size(100);
```

syslog_server( )

Use this method to set or retrieve the syslog server of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::Grid::SyslogServer object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get syslog server
my $syslog_server = $grid_member->syslog_server();
#Modify syslog server
my $syslog1 = Infoblox::Grid::SyslogServer->new(
    ipv4addr => "44.44.44.44",
    severity => "debug",
);
my $syslog2 = Infoblox::Grid::SyslogServer->new(
    ipv4addr => "55.55.55.55",
    severity => "debug",
);

$grid_member->syslog_server([$syslog1, $syslog2]);
#Undef syslog server
$grid_member->syslog_server(undef);
```

time_zone( )

Use this method to set or retrieve the time zone of the grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The UTC string that represents the time zone. For example "(UTC - 5:00) Eastern Time (US and Canada)". Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get time zone
my $time_zone = $grid_member->time_zone();
#Modify time zone
$grid_member->time_zone("(UTC - 5:00) Eastern Time (US and Canada)");
#Undef time zone
```

```
$grid_member->time_zone(undef);
```

trap_comm_string( )

Use this method to set or retrieve the Simple Network Management Protocol (SNMP) trap community string of the grid member. An SNMP trap community string is a text string that acts as a password. It is used to authenticate messages that are sent between the grid member and the SNMP management systems.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the Simple Network Management Protocol (SNMP) trap community, with a maximum of 50 bytes. Default value is undefined. To undefine trap_comm_string we also need to undefine "trap_receiver" attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get trap community string
my $trap_comm_string = $grid_member->trap_comm_string();
#Modify trap community string
$grid_member->trap_comm_string("trap_string");
```

trap_receiver( )

Use this method to set or retrieve the Simple Network Management Protocol (SNMP) trap receivers of a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 addresses and the Infoblox::Grid::SNMP::TrapReceiver manpage objects. Default value is undefined. To undefine trap_receiver, you must also undefine the "trap_comm_string" attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get trap receiver
my $trap_receiver = $grid_member->trap_receiver();
#Modify trap receiver (SNMPv2)
$grid_member->trap_receiver(["10.0.0.10"]);
#SNMPv3 trap receivers
my $receiver1 = Infoblox::Grid::SNMP::TrapReceiver->new( ip_addr => "19.20.21.22", user => $user);
$gridmember->trap_receiver([$receiver1]);
```

vpn_on_mgmt( )

Use this method to set or retrieve the VPN services on the MGMT port of a grid member that is a single appliance.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Note that node1_vpn_on_mgmt, node2_vpn_on_mgmt and vpn_on_mgmt share the value; setting any of them will have the same effect.

Parameter

Specify "true" to set the vpn_on_mgmt flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get VPN on MGMT port flag
my $vpn_on_mgmt = $grid_member->vpn_on_mgmt();
#Modify VPN on MGMT port flag
```

```
$grid_member->vpn_on_mgmt("false");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, get, modify, search and remove. This sample also includes error handling for the operations.

#Preparation prior to getting and modifying a grid member object

```
use strict;
use Infoblox;

#refers to Infoblox Appliance IP address
my $host_ip = "192.168.1.2";

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
);
unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $sysserv1 = Infoblox::Grid::SyslogServer->new(
    ipv4addr => "7.7.7.7"
);
unless ($sysserv1) {
       die("Construct sysserv1 failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Sysserv1 created successfully\n";
```

#Create a grid member object

```
my $grid_member = Infoblox::Grid::Member->new(
    name          => "hostname.com",
    ipv4addr      => "3.0.0.2",
    mask          => "255.0.0.0",
    gateway       => "3.0.0.1",
    syslog_server => [$sysserv1],
    comment       => "Script Generated"
    );
unless ($grid_member) {
       die("Construct grid member object failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Grid member object created successfully\n";

#Adding the member object to appliance through session
my $response = $session->add( $grid_member );

unless ($response) {
    die("Add grid member failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print"Grid member added successfully \n";
```

#Search for a grid member object

```
 my @retrieved_objs = $session->search(
     object => "Infoblox::Grid::Member",
     name   => ".*\\.com");
 my $object = $retrieved_objs[0];

 unless ($object) {
        die("Search grid member object failed: ",
                $session->status_code() . ":" . $session->status_detail());
 }
 print "Search grid member object found at least 1 matching entry\n";
```

#Get and modify the member object

```
#Get grid member through session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member",
    name   => "hostname.com"
    );
my $grid_member = $retrieved_objs[0];

unless ($grid_member) {
    die("Get grid member object failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print"Get grid member successful \n";

# Modify one of the attributes of selected grid member.

$grid_member->comment("Modified member object");

#Applying the changes
$session->modify($grid_member)
   or die("Modify grid member failed",
            $session->status_code() . ":" . $session->status_detail());
print"Grid member modified successfully \n";
```

#Remove the grid member object

```
#Get grid member through session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member",
    name   => "hostname.com"
    );
my $desired_grid_member = $retrieved_objs[0];

unless ($desired_grid_member) {
    die("Get grid member object failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print"Get grid member successful \n";

#Submit the object for removal
$session->remove( $desired_grid_member )
    or die("Remove grid member failed",
           $session->status_code() . ":" . $session->status_detail());
print"Grid member removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::Member, Infoblox::DNS::Nameserver, Infoblox::Grid, Infoblox::Grid::NTPKey, Infoblox::Grid::SNMP::Admin, Infoblox::Grid::SyslogServer, Infoblox::Session, Infoblox::Grid::Member::OSPF

COPYRIGHT

Infoblox::Grid::Member::BGP::AS - BGP object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::Member->bgp_as( )
- METHODS
  - as( )
  - keepalive( )
  - holddown( )
  - neighbors( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::BGP::AS - BGP (Border Gateway Protocol) Autonomous System (AS) object.

DESCRIPTION

An Infoblox::BGP::AS object represents a BGP Autonomous System configured at the grid member level.

CONSTRUCTOR

```
my $bgp_as = Infoblox::Grid::Member::BGP::AS->new(
    as              => $num,                          # Optional / Default value is undefined
    keepalive       => $num,                          # Optional / Default value is 4
    holddown        => $num,                          # Optional / Default value is 16
    neighbors       => [ $neighbor1, $neighbor2, ... ], # Optional / Default value is empty
);
```

MODULE METHODS

The following functions can be applied to a BGP AS object.

Infoblox::Grid::Member->bgp_as( )

Use this function to specify BGP Autonomous Systems for the grid member. See Infoblox::Grid::Member->bgp_as() for parameters and return values.

Example

```
#Construct a BGP neighbor
my $neighbor = Infoblox::Grid::Member::BGP::Neighbor->new(
    neighbor_ip          => "172.16.0.1",
    remote_as           => 1234,
    interface           => "LAN_HA",
);

#Construct a BGP AS
my $autonomous_system = Infoblox::Grid::Member::BGP::AS->new(
    as          => 269,
    keepalive       => 4,
    holddown        => 16,
    neighbors       => [ $neighbor ]
);

# Configure BGP on the Grid Member object
my $response = $Grid_Member->bgp_as([$as]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of an Infoblox::BGP::AS object.

as( )

Use this method to set or retrieve the number of this AS.

A valid AS number is required to advertise to neighbors defined in this AS.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An AS number is a 16-bit integer from 1 to 65535.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get AS number
my $as_number = $autonomous_system->as();
#Modify AS number
$autonomous_system->as(300);
```

keepalive( )

Use this method to set or retrieve the AS keepalive timer.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of seconds for the keepalive timer. The valid value is from 1 to 21845 seconds.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get AS keepalive timer
my $keepalive = $autonomous_system->keepalive();
#Modify AS keepalive timer
$autonomous_system->keepalive(3600);
```

holddown( )

Use this method to set or retrieve the AS hold down timer.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The number of seconds for the hold down timer. The valid value is from 3 to 65535 seconds.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get Autonomous System hold down timer
my $holddown = $autonomous_system->holddown();
#Modify Autonomous System hold down timer
$autonomous_system->holddown(3600);
```

neighbors( )

Use this method to set or retrieve the BGP neighbors for the Infoblox::BGP::AS object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::Grid::Member::BGP::Neighbor object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get AS neighbor list
my $neighbors = $autonomous_system->neighbors();
#Modify AS neighbors
$autonomous_system->neighbors( [ $neighbor1, $neighbor2 ] );
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member, Infoblox::Grid::Member::BGP::Neighbor

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::BGP::Neighbor - BGP neighbor object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::Member::BGP::AS->neighbors( )
- METHODS
  - interface( )
  - neighbor_ip( )
  - remote_as( )
  - comment( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::BGP::Neighbor - BGP (Border Gateway Protocol) neighbor object

DESCRIPTION

An Infoblox::BGP::Neighbor object represents the BGP neighbor that is configured in an Autonomous System (AS). BGP neighbors are configured at the grid member level.

CONSTRUCTOR

```
my $neighbor = Infoblox::Grid::Member::BGP::Neighbor->new(
    interface           => $string                              #Optional / default is "LAN_HA"
    neighbor_ip         => $ipv4addr                            #Required
    remote_as           => $num                                 #Required
    comment             => $string                              #Optional / default is undefined
);
```

MODULE METHODS

The following functions can be applied to a BGP AS object.

Infoblox::Grid::Member::BGP::AS->neighbors( )

Use this function to specify BGP Autonomous Systems for the grid member. See Infoblox::Grid::Member::BGP::AS->neighbors() for parameters and return values.

Example

```
#Construct a BGP neighbor
my $neighbor = Infoblox::Grid::Member::BGP::Neighbor->new(
    neighbor_ip          => "172.16.0.1",
    remote_as           => 1234,
    interface           => "LAN_HA",
);

#Construct a BGP AS
my $autonomous_system = Infoblox::Grid::Member::BGP::AS->new(
    as          => 269,
    keepalive       => 4,
    holddown        => 16,
    neighbors       => [ $neighbor ]
);

# Configure BGP on the Grid Member object
my $response = $Grid_Member->bgp_as([$as]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of an Infoblox::BGP::AS object.

interface( )

Use this method to set or retrieve the interface that sends BGP advertisement information.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the interface in string format. The only supported value is "LAN_HA".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get interface
my $interface = $neighbor->interface();
#Modify interface
$neighbor->interface("LAN_HA");
```

neighbor_ip( )

Use this method to set or retrieve the IP address of the BGP neighbor.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The IPv4 address of the BGP neighbor.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get neighbor IP address
my $neighbor_ip = $neighbor->neighbor_ip();
#Modify neighbor IP address
$neighbor->neighbor_ip("192.168.1.2");
```

remote_as( )

Use this method to set or retrieve the remote AS number of the BGP neighbor.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An AS number is a 16-bit number from 1 to 65535.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get neighbor remote AS number
my $remote_as = $neighbor->remote_as();
#Modify remote AS number
$neighbor->remote_as(269);
```

comment( )

Use this method to set or retrieve a descriptive comment.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get neighbor comment
my $comment = $neighbor->comment();
#Modify comment
$neighbor->comment("my comment");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

L<Infoblox::Grid::Member::BGP::AS|Infoblox::Grid::Member::BGP::AS>

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::CaptivePortal - Captive Portal object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - authn_server_group ( )
  - company_name ( )
  - enable_syslog_auth_failure ( )
  - enable_syslog_auth_success ( )
  - enable_user_type ( )
  - encryption ( )
  - files ( )
  - guest_custom_field1_name ( )
  - guest_custom_field1_required ( )
  - guest_custom_field2_name ( )
  - guest_custom_field2_required ( )
  - guest_custom_field3_name ( )
  - guest_custom_field3_required ( )
  - guest_custom_field4_name ( )
  - guest_custom_field4_required ( )
  - guest_email_required ( )
  - guest_first_name_required ( )
  - guest_last_name_required ( )
  - guest_middle_name_required ( )
  - guest_phone_required ( )
  - helpdesk_message ( )
  - listen_address ( )
  - name ( )
  - network_view ( )
  - port ( )
  - service_enabled ( )
  - syslog_auth_failure_level ( )
  - syslog_auth_success_level ( )
  - welcome_message ( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::CaptivePortal - Captive Portal object.

DESCRIPTION

This object represents the captive portal configuration.

CONSTRUCTOR

The Infoblox::Grid::Member::CaptivePortal object is automatically generated upon the successful installation of the Infoblox appliance. It does not require manual construction.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a Captive Portal object.

Infoblox::Session->get( )

Use this method to retrieve matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attribute to get a specific Captive Portal object:

 name - Required. A domain name in string format.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member::CaptivePortal",
    name   => "infoblox.localdomain"
    );
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use this method to modify the welcome message.
$captive_portal->welcome_message("New welcome message");
#Submit modification
my $response = $session->modify( $captive_portal );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of the captive portal.

authn_server_group ( )

Use this method to set or retrieve the authentication server group assigned to this captive portal.

Parameter

The valid values are an Infoblox::Grid::Admin::AdAuthService or an Infoblox::Grid::Admin::RadiusAuthService object. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting authn_server_group
my $authn_server_group = $captive_portal->authn_server_group();
#Modifying authn_server_group
$captive_portal->authn_server_group($adservice);
```

company_name ( )

Use this method to set or retrieve the company name that appears in the guest registration page.

Parameter

The valid value is a string. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting company_name
my $company_name = $captive_portal->company_company_name();
#Modifying company_name
$captive_portal->company_name('Company Name');
```

enable_syslog_auth_failure ( )

Use this method to set or retrieve the flag that controls whether authentication failures are logged to syslog.

Parameter

Specify "true" to enable logging or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_syslog_auth_failure
```

```
my $enable_syslog_auth_failure = $captive_portal->enable_syslog_auth_failure();
#Modifying enable_syslog_auth_failure
$captive_portal->enable_syslog_auth_failure('true');
```

enable_syslog_auth_success ( )

Use this method to set or retrieve the flag that controls whether successful authentications are logged to syslog.

Parameter

Specify "true" to enable logging or "false" to disable it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_syslog_auth_success
my $enable_syslog_auth_success = $captive_portal->enable_syslog_auth_success();
#Modifying enable_syslog_auth_success
$captive_portal->enable_syslog_auth_success('false');
```

enable_user_type ( )

Use this method to set or retrieve the type of user to be enabled for the captive portal.

Parameter

Valid values are "Authenticated", "Guest" or "Both". The default value is "Authenticated".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_user_type
my $enable_user_type = $captive_portal->enable_user_type();
#Modifying enable_user_type
$captive_portal->enable_user_type('Both');
```

encryption ( )

Use this method to set or retrieve the encryption the captive portal uses.

Parameter

Valid values are 'None' or 'SSL'. The default value is 'None'.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting encryption
my $encryption = $captive_portal->encryption();
#Modifying encryption
$captive_portal->encryption('SSL');
```

files ( )

Use this method to set or retrieve the list of files associated with the captive portal.

Parameter

The valid value is an array reference that contains Infoblox::Grid::Member::CaptivePortal::File object(s). New files must be added using Infoblox::Session->import_data(). However, if a file is removed from this list, it is also removed from the captive portal.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting files
my $files = $captive_portal->files();
#Modifying files
$captive_portal->files([$logo]);
```

guest_custom_field1_name ( )

Use this method to set or retrieve the name of the custom field that you are adding to the guest registration page.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field1_name
my $guest_custom_field1_name = $captive_portal->guest_custom_field1_guest_custom_field1_name();
#Modifying guest_custom_field1_name
$captive_portal->guest_custom_field1_name('Extra field 1');
```

guest_custom_field1_required ( )

Use this method to set or retrieve the flag that controls whether the custom field is required.

Parameter

Specify "true" to set the flag or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field1_required
my $guest_custom_field1_required = $captive_portal->guest_custom_field1_required();
#Modifying guest_custom_field1_required
$captive_portal->guest_custom_field1_required('true');
```

guest_custom_field2_name ( )

Use this method to set or retrieve the name of the custom field that you are adding to the guest registration page.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field2_name
my $guest_custom_field2_name = $captive_portal->guest_custom_field2_guest_custom_field2_name();
#Modifying guest_custom_field2_name
$captive_portal->guest_custom_field2_name('Extra field 2');
```

guest_custom_field2_required ( )

Use this method to set or retrieve the flag that controls whether the custom field is required.

Parameter

Specify "true" to set the flag or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field2_required
my $guest_custom_field2_required = $captive_portal->guest_custom_field2_required();
#Modifying guest_custom_field2_required
$captive_portal->guest_custom_field2_required('true');
```

guest_custom_field3_name ( )

Use this method to set or retrieve the name of the custom field that you are adding to the guest registration page.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field3_name
my $guest_custom_field3_name = $captive_portal->guest_custom_field3_guest_custom_field3_name();
#Modifying guest_custom_field3_name
$captive_portal->guest_custom_field3_name('Extra field 3');
```

guest_custom_field3_required ( )

Use this method to set or retrieve the flag that controls whether the custom field is required.

Parameter

Specify "true" to set the flag or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field3_required
my $guest_custom_field3_required = $captive_portal->guest_custom_field3_required();
#Modifying guest_custom_field3_required
$captive_portal->guest_custom_field3_required('true');
```

guest_custom_field4_name ( )

Use this method to set or retrieve the name of the custom field that you are adding to the guest registration page.

Parameter

The valid value is a string. The default value is undefined. The maximum length is 200 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field4_name
my $guest_custom_field4_name = $captive_portal->guest_custom_field4_guest_custom_field4_name();
#Modifying guest_custom_field4_name
$captive_portal->guest_custom_field4_name('Extra field 4');
```

guest_custom_field4_required ( )

Use this method to set or retrieve the flag that controls whether the custom field is required.

Parameter

Specify "true" to set the flag or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_custom_field4_required
my $guest_custom_field4_required = $captive_portal->guest_custom_field4_required();
#Modifying guest_custom_field4_required
$captive_portal->guest_custom_field4_required('true');
```

guest_email_required ( )

Use this method to set or retrieve the flag that controls whether the email address of the guest is required.

Parameter

Specify "true" to set this flag or "false" to disable it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_email_required
my $guest_email_required = $captive_portal->guest_email_required();
#Modifying guest_email_required
$captive_portal->guest_email_required('false');
```

guest_first_name_required ( )

Use this method to set or retrieve the flag that controls whether the first name of the guest is required.

Parameter

Specify "true" to enable or "false" to disable it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_first_name_required
my $guest_first_name_required = $captive_portal->guest_first_name_required();
#Modifying guest_first_name_required
$captive_portal->guest_first_name_required('false');
```

guest_last_name_required ( )

Use this method to set or retrieve the flag that controls whether the last name of the guest is required.

Parameter

Specify "true" to set the flag or "false" to disable it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_last_name_required
my $guest_last_name_required = $captive_portal->guest_last_name_required();
#Modifying guest_last_name_required
$captive_portal->guest_last_name_required('false');
```

guest_middle_name_required ( )

Use this method to set or retrieve the flag that controls whether the middle name of the guest is required.

Parameter

Specify "true" to set the flag or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_middle_name_required
my $guest_middle_name_required = $captive_portal->guest_middle_name_required();
#Modifying guest_middle_name_required
$captive_portal->guest_middle_name_required('true');
```

guest_phone_required ( )

Use this method to set or retrieve the flag that controls whether the phone number of the guest is required.

Parameter

Specify "true" to set this flag or "false" to disable it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting guest_phone_required
my $guest_phone_required = $captive_portal->guest_phone_required();
#Modifying guest_phone_required
$captive_portal->guest_phone_required('false');
```

helpdesk_message ( )

Use this method to set or retrieve the helpdesk message that appears in the guest registration page.

Parameter

The valid value is a string. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting helpdesk_message
my $helpdesk_message = $captive_portal->helpdesk_message();
#Modifying helpdesk_message
$captive_portal->helpdesk_message('In case of problems, please contact us at 555-555-5555');
```

listen_address ( )

Use this method to set or retrieve the IP address on which the captive portal listens.

Parameter

The valid values are 'VIP', 'LAN2' or an IPv4 address. The default value is 'VIP'.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting listen_address
my $listen_address = $captive_portal->listen_address();
#Modifying listen_address
$captive_portal->listen_address('VIP');
```

name ( )

Use this method to retrieve the hostname of the grid member that hosts the captive portal.

Returns

The method returns the attribute value.

Example

```
#Getting name
my $name = $captive_portal->name();
```

network_view ( )

Use this method to set or retrieve the network view of the captive portal.

Parameter

The valid value is a network view name. The default is the system default network view.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting network_view
my $network_view = $captive_portal->network_view();
#Modifying network_view
$captive_portal->network_view('otherview');
```

port ( )

Use this method to set or retrieve the TCP port used by the Captive Portal service. The port is required when the Captive Portal service is enabled.

Parameter

Valid values are between 1 and 63999. The default value is 4433. Please note that setting the port number to 80 or 443 might impact performance.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting the port
my $port = $captive_portal->port();
#Modifying the port
$captive_portal->port(4457);
```

service_enabled ( )

Use this method to set or retrieve the state of the captive portal service.

Parameter

Specify "true" to enable the service or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting service_enabled
my $service_enabled = $captive_portal->service_enabled();
#Modifying service_enabled
$captive_portal->service_enabled('false');
```

syslog_auth_failure_level ( )

Use this method to set or retrieve the syslog level at which authentication failures are logged.

Parameter

Valid values are "debug", "info", "notice", "warning", "err", "crit", "alert" or "emerg" . The default value is "info".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting syslog_auth_failure_level
my $syslog_auth_failure_level = $captive_portal->syslog_auth_failure_level();
#Modifying syslog_auth_failure_level
$captive_portal->syslog_auth_failure_level('notice');
```

syslog_auth_success_level ( )

Use this method to set or retrieve the syslog level at which successful authentications are logged.

Parameter

Valid values are "debug", "info", "notice", "warning", "err", "crit", "alert" or "emerg" . The default value is "info".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting syslog_auth_success_level
my $syslog_auth_success_level = $captive_portal->syslog_auth_success_level();
#Modifying syslog_auth_success_level
$captive_portal->syslog_auth_success_level('warning');
```

welcome_message ( )

Use this method to set or retrieve the welcome message that appears in the guest registration page.

Parameter

The valid value is a string. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting welcome_message
my $welcome_message = $captive_portal->welcome_message();
#Modifying welcome_message
$captive_portal->welcome_message('Welcome to our network!');
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to a captive portal object, such as get and modify an object. This sample code also includes error handling for the operations.

#Preparation prior to operations on a captive portal object

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

# Upload a new AUP to the captive portal, from now on
# the files() method will contain an Infoblox::Grid::Member::CaptivePortal::File
# object referencing this.
$session->import_data(
                      type    => 'captive_portal_file',
```

```
                        path    => 'aup.txt',
                        member  => '192.168.1.2',
                        cp_type => 'aup'
 )
 or die("file import failed: ",
        $session->status_code(), $session->status_detail());

 my $ads1 = Infoblox::Grid::Admin::RadiusAuthServer->new(
                                                        comment  => 'server 1',
                                                        fqdn_or_ip => '10.0.1.1',
                                                        shared_secret => 'secret1',
                                                       );

 my $ads2 = Infoblox::Grid::Admin::RadiusAuthServer->new(
                                                        comment  => 'server 2',
                                                        fqdn_or_ip => 'domain.com',
                                                        shared_secret => 'secret2',
                                                      );

#Create the RADIUS Auth Service.

 my $service = Infoblox::Grid::Admin::RadiusAuthService->new(
                                                            acct_retries   => 2000,
                                                            auth_retries   => 9,
                                                            cache_ttl      => 500,
                                                            disabled       => 'true',
                                                            enable_cache   => 'true',
                                                            recovery_ttl   => 50,
                                                            selection_mode => 'ROUND_ROBIN',
                                                            acct_timeout   => 1100,
                                                            auth_timeout   => 1200,
                                                            name           => 'some.name.com',
                                                            radius_servers => [$ads1, $ads2]
                                                         );

 unless($service){
       die("Construct service object failed: ",
              Infoblox::status_code(). ":" .Infoblox::status_detail());
       }
 print "service object object created successfully.\n";

 $session->add($service)
     or die("Add service object failed: ",
              $session->status_code(). ":" .$session->status_detail());

 print"Service object added successfully.\n";

#Get and modify the captive portal property object.

 #Get the captive portal object. Considering the id grid member named infoblox.localdomain was previously
created.
 my @object = $session->get(
     object => "Infoblox::Grid::Member::CaptivePortal",
     name   => "infoblox.localdomain"
     )
     or die("get captive portal failed: ",
          $session->status_code(), $session->status_detail());

 #Modify captive portal object.
 unless (scalar(@object) == 0) {
     my $object = $object[0];

     if ($object) {
         $object->authn_server_group($service);
         $object->guest_phone_required('false');
         $object->guest_custom_field1_name('Custom field 1');
         $object->guest_custom_field1_required('true');
         $object->guest_custom_field2_name('Custom field 2');
         $object->guest_custom_field2_required('true');

         # This will remove the AUP uploaded above
         $object->files([]);

         #Applying the changes
         $session->modify($object)
             or die("Modify captive portal failed: ",
                  $session->status_code(), $session->status_detail());

     }
      } else {
```

```
    print "No grid captive portal found.";
     }

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member::CaptivePortal::File, Infoblox::Grid::Admin::AdAuthService, Infoblox::Grid::Admin::RadiusAuthService, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::CaptivePortal::File - Captive Portal File object

- NAME
- DESCRIPTION
- MODULE METHODS
  - Infoblox::Grid::Member::CaptivePortal->files( )
- METHODS
  - name( )
  - type( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::CaptivePortal::File - Captive Portal File object

DESCRIPTION

This object is used to represent files uploaded to the captive portal. All methods are read-only.

MODULE METHODS

The following functions can be applied to a CaptivePortal::File object.

Infoblox::Grid::Member::CaptivePortal->files( )

Use this function to retrieve a reference to an array of Infoblox::Grid::Member::CaptivePortal::File objects that represent the files uploaded to the captive portal.

Example

```
#Retrieve the captive portal object
my $member = $session->get(object => 'Infoblox::Grid::Member::CaptivePortal', name =>
'cportal.com');

# Access its files
my @files = @{$member->files()};
print $files[0]->name();
```

METHODS

This section describes all the methods that you can use to retrieve the attribute value of a file uploaded to a captive portal.

name( )

Use this method to retrieve the name of the uploaded file.

Returns

The method returns the attribute value.

Example

```
#Get name
my $name = $service_status->name();
```

type( )

Use this method to retrieve the type of the uploaded file.

Returns

The method returns the attribute value. Valid values are "Logo", "Footer", "Header", and "AUP"

Example

```
#Get type
my $type = $service_type->type();
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member::CaptivePortal/>

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::DHCP - DHCP member object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - auth_server_group( )
  - authority( )
  - authn_captive_portal
  - authn_captive_portal_authenticated_filter
  - authn_captive_portal_enabled
  - authn_captive_portal_guest_filter
  - auth_server_group_enabled
  - bootfile( )
  - bootserver( )
  - ddns_generate_hostname( )
  - ddns_retry_interval( )
  - ddns_server_always_updates( )
  - ddns_ttl( )
  - ddns_update_fixed_addresses( )
  - ddns_use_option81( )
  - deny_bootp( )
  - email_list( )
  - enable_ddns( )
  - enable_dhcp( )
  - enable_dhcpv6_service( )
  - enable_dhcp_on_lan2( )
  - enable_dhcp_thresholds( )
  - enable_email_warnings( )
  - enable_gss_tsig( )
  - enable_ifmap( )
  - enable_snmp_warnings( )
  - extensible_attributes( )
  - gss_tsig_key( )
  - high_water_mark( )
  - ifmap_password( )
  - ifmap_username( )
  - ignore_dhcp_option_list_request( )
  - ipv6_ddns_enable_option_fqdn( )
  - ipv6_ddns_server_always_updates( )
  - ipv6_ddns_domainname( )
  - ipv6_ddns_hostname( )
  - ipv6_ddns_ttl( )
  - ipv6_domain_name( )
  - ipv6_domain_name_servers( )
  - ipv6_enable_ddns( )
  - ipv6_enable_gss_tsig( )
  - ipv6_enable_retry_updates( )
  - ipv6_generate_hostname( )
  - ipv6_gss_tsig_key( )
  - ipv6_kdc_server( )
  - ipv6_microsoft_code_page( )
  - ipv6_options( )
  - ipv6_recycle_leases( )
  - ipv6_retry_updates_interval( )
  - ipv6_server_duid( )
  - ipv6_update_dns_on_lease_renewal( )
  - kdc_server( )
  - lease_scavenge_time( )
  - log_lease_events( )
  - low_water_mark( )
  - name( )
  - nextserver( )
  - microsoft_code_page( )
  - options( )
  - option60_match_rules( )
  - override_ddns_ttl( )
  - override_enable_dhcpv6_service( )
  - override_ipv6_ddns_domainname( )

- override_ipv6_ddns_hostname( )
- override_ipv6_ddns_enable_option_fqdn( )
- override_ipv6_ddns_ttl( )
- override_ipv6_domain_name( )
- override_ipv6_domain_name_servers( )
- override_ipv6_enable_ddns( )
- override_ipv6_enable_gss_tsig( )
- override_ipv6_enable_retry_updates( )
- override_ipv6_generate_hostname( )
- override_ipv6_microsoft_code_page( )
- override_ipv6_options( )
- override_ipv6_recycle_leases( )
- override_ipv6_update_dns_on_lease_renewal( )
- override_lease_scavenge_time( )
- override_log_lease_events( )
- override_preferred_lifetime( )
- override_update_dns_on_lease_renewal( )
- override_valid_lifetime( )
- ping_count( )
- ping_timeout( )
- preferred_lifetime( )
- pxe_lease_time( )
- recycle_leases( )
- retry_ddns_updates( )
- syslog_facility( )
- update_dns_on_lease_renewal( )
- valid_lifetime( )

- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::DHCP - DHCP member object.

DESCRIPTION

When you configure an Infoblox appliance to function as a DHCP server, you can set certain properties that control how the server operates. You can also specify the configuration information the server includes in its DHCP messages. When a DHCP server assigns an IP address to a client, it can include configuration information the client needs to connect to the network and communicate with other hosts and appliances in the network.

Though you can set general DHCP properties at the grid level, you enable DHCP services at the member level only. Infoblox recommends that you configure the DHCP parameters before you enable DHCP on the Infoblox appliance.

CONSTRUCTOR

The Infoblox::Grid::Member::DHCP object is automatically generated upon the successful installation of the Infoblox appliance. It does not require manual construction.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a member DHCP object.

Infoblox::Session->get( )

Use this method to retrieve matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attribute to get a specific member DHCP object:

 name - Required. A domain name in string format.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member::DHCP",
    name   => "infoblox.localdomain"
```

```
    );

# get all Grid::Member::DHCP objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member::DHCP",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify dhcp service.
$member_dhcp->enable_dhcp("true");
#Submit modification
my $response = $session->modify( $member_dhcp );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of the DHCP properties of a grid member.

auth_server_group( )

Use this method to set or retrieve the Authentication Server Group object associated with this member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An Infoblox::Grid::Admin::RadiusAuthService object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting auth_server_group

my $auth_server_group = $memberdhcp->auth_server_group( );
#Modifying auth_server_group
$auth_server_group->$memberdhcp($radius_service);
```

authority( )

Use this method to set or retrieve the authority flag of a grid member. This flag specifies if a DHCP server is authoritative for a domain.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the authority flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authority
my $authority = $memberdhcp->authority();
#Modify authority
$memberdhcp->authority("true");
```

authn_captive_portal

The captive portal responsible for authentications for this DHCP member.

Parameter

The valid value is a name in FQDN format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authn_captive_portal
my $authn_captive_portal = $memberdhcp->authn_captive_portal();
#Modify authn_captive_portal
$memberdhcp->authn_captive_portal("cpmember.com");
```

authn_captive_portal_authenticated_filter

The MAC filter representing the authenticated range.

Parameter

The valid value is an Infoblox::DHCP::Filter::MAC object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authn_captive_portal_authenticated_filter
my $authn_captive_portal_authenticated_filter = $memberdhcp-
>authn_captive_portal_authenticated_filter();
#Modify authn_captive_portal_authenticated_filter
$memberdhcp->authn_captive_portal_authenticated_filter($macfilter);
```

authn_captive_portal_enabled

Use this method to set or retrieve the flag that controls if this DHCP member is enabled for captive portal authentication.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authn_captive_portal_enabled
my $authn_captive_portal_enabled = $memberdhcp->authn_captive_portal_enabled();
#Modify authn_captive_portal_enabled
$memberdhcp->authn_captive_portal_enabled("true");
```

authn_captive_portal_guest_filter

The MAC filter representing the guest range.

Parameter

The valid value is an Infoblox::DHCP::Filter::MAC object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authn_captive_portal_guest_filter
my $authn_captive_portal_guest_filter = $memberdhcp->authn_captive_portal_guest_filter();
#Modify authn_captive_portal_guest_filter
$memberdhcp->authn_captive_portal_guest_filter($macfilter);
```

auth_server_group_enabled

Use this method to set or retrieve the flag that controls if this DHCP member can send authentication requests to an authentication server group.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get auth_server_group_enabled
my $auth_server_group_enabled = $memberdhcp->auth_server_group_enabled();
#Modify auth_server_group_enabled
$memberdhcp->auth_server_group_enabled("true");
```

bootfile( )

Use this method to set or retrieve the name of a file that DHCP clients need to boot. This setting overrides the grid level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::Grid::Member::DHCP->bootserver() and Infoblox::Grid::Member::DHCP->nextserver().

Parameter

The name of the file that the client must download. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootfile
my $bootfile = $memberdhcp->bootfile();
#Modify bootfile
$memberdhcp->bootfile("bootfile.txt");
#Un-override bootfile
$memberdhcp->bootfile(undef);
```

bootserver( )

Use this method to set or retrieve the name of the server on which a boot file is stored. This setting overrides the grid level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::Grid::Member::DHCP->bootfile() and Infoblox::Grid::Member::DHCP->nextserver().

Parameter

The boot server IP address or name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot server name can have a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bootserver
my $bootserver = $memberdhcp->bootserver();
#Modify bootserver
$memberdhcp->bootserver("hostname.com");
#Un-override bootserver
$memberdhcp->bootserver(undef);
```

ddns_generate_hostname( )

Use this method to set or retrieve the ddns_generate_hostname value. Set this value to enable/disable the ability of a member DHCP server to generate a host name and update DNS with this host name when it receives a DHCP REQUEST message that does not include a host name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_generate_hostname
my $ddns_generate_hostname = $memberdhcp->ddns_generate_hostname();
#Modify disable
$memberdhcp->ddns_generate_hostname("true");
```

ddns_retry_interval( )

Use this method to set or retrieve the ddns_retry_interval value. Set the retry interval when the member DHCP server makes repeated attempts to send DDNS updates to a DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the retry interval in minutes. The default retry interval is five minutes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_retry_interval
my $ddns_retry_interval = $memberdhcp->ddns_retry_interval();
#Modify ddns_retry_interval
$memberdhcp->ddns_retry_interval(10);
```

ddns_server_always_updates( )

Use this method to set or retrieve the ddns_server_always_updates value of a member DHCP server. Select this check box to allow only the DHCP server to update DNS,regardless of the requests from the DHCP clients. This setting overrides the grid level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set ddns_server_always_updates flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_server_always_updates
my $ddns_server_always_updates = $memberdhcp->ddns_server_always_updates();
#Modify ddns_server_always_updates
$memberdhcp->ddns_server_always_updates("true");
```

ddns_ttl( )

Use this method to set or retrieve the ddns_ttl value of a member DHCP server. The DDNS TTL (Dynamic DNS Time To Live) value specifies the number of seconds an IP address for the name is cached.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ddns_ttl method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ddns_ttl attribute to "false".

Note that when ddns_ttl is set to a defined value and override_dddns_ttl is set to "false", the last operation takes precedence. Thus the sequence $object->ddns_ttl(12); $object->override_ddns_ttl("false"); will set override_ddns_ttl to "false", and the sequence $object->override_ddns_ttl("false"); $object->ddns_ttl(12); will result in override_ddns_ttl="true".

Parameter

A 32-bit integer (range from 0 to 4294967295) that specifies the number of seconds the IP address for the name is cached. Default value is half of lease time.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_ttl
my $ddns_ttl = $memberdhcp->ddns_ttl();
#Modify ddns_ttl
$memberdhcp->ddns_ttl(1000);
```

ddns_update_fixed_addresses( )

Use this method to set or retrieve the ddns_update_fixed_addresses value. Set this value to enable/disable the member DHCP server to update the A and PTR record of clients with a fixed address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the member DHCP server to update the A and PTR record of clients with a fixed address or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_update_fixed_addresses
my $ddns_update_fixed_addresses = $memberdhcp->ddns_update_fixed_addresses();
#Modify ddns_update_fixed_addresses
$memberdhcp->ddns_update_fixed_addresses("true");
```

ddns_use_option81( )

Use this method to set or retrieve the ddns_use_option81 value of a member DHCP server. Set this value to enable/disable support for option 81.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable support for option 81 or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_use_option81
my $ddns_use_option81 = $memberdhcp->ddns_use_option81();
#Modify ddns_use_option81
$memberdhcp->ddns_use_option81("true");
```

deny_bootp( )

Use this method to set or retrieve the deny_bootp value of a member DHCP server. Set this value to enable/disable deny_bootp and Grid level.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable deny_bootp or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get deny_bootp
my $deny_bootp = $memberdhcp->deny_bootp();
#Modify deny_bootp
$memberdhcp->deny_bootp("true");
```

email_list( )

Use this method to set or retrieve the email_list value of a member DHCP server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference with the e-mail values.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get email_list
my $email_list = $memberdhcp->email_list();
#Modify email_list
$memberdhcp->email_list(["admin\@infoblox.com"]);
```

enable_ddns( )

Use this method to set or retrieve the enable_ddns value of a member DHCP server. Set this value to enable/disable the member DHCP server to send DDNS updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable support for DDNS updates or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ddns
my $enable_ddns= $memberdhcp->enable_ddns();
#Modify enable_ddns
$memberdhcp->enable_ddns("true");
```

enable_dhcp( )

Use this method to set or retrieve the enable_dhcp value of a member. Set this value to enable/disable DHCP service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable DHCP or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcp
my $enable_dhcp= $memberdhcp->enable_dhcp();
#Modify enable_dhcp
$memberdhcp->enable_dhcp("true");
```

enable_dhcpv6_service( )

Use this method to set or retrieve the enable_dhcpv6_service flag. Set this flag to enable DHCPv6 service for the member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_enable_dhcpv6_service method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_enable_dhcpv6_service attribute to "false".

Note that when enable_dhcpv6_service is set to a defined value and override_denable_dhcpv6_service is set to "false", the last operation takes precedence. Thus the sequence $object->enable_dhcpv6_service("true"); $object->override_enable_dhcpv6_service("false"); will set override_enable_dhcpv6_service to "false", and the sequence $object->override_enable_dhcpv6_service("false"); $object->enable_dhcpv6_service("true"); will result in override_enable_dhcpv6_service="true".

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcpv6_service
my $enable_dhcpv6_service = $memberdhcp->enable_dhcpv6_service();
#Modify enable_dhcpv6_service
$memberdhcp->enable_dhcpv6_service("true");
```

enable_dhcp_on_lan2( )

Use this method to set or retrieve the enable_dhcp_on_lan2 value of a member. Set this value to enable/disable DHCP service on the LAN2 interface.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable DHCP on the LAN2 interface or "false" to disable it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcp_on_lan2
my $enable_dhcp_on_lan2 = $memberdhcp->enable_dhcp_on_lan2();
#Modify enable_dhcp_on_lan2
$memberdhcp->enable_dhcp_on_lan2("true");
```

enable_dhcp_thresholds( )

Use this method to set or retrieve the enable_dhcp_thresholds value of a member DHCP server. Thresholds represents the watermarks above or below which address usage in a network is unexpected and might warrant your attention. This setting overrides the grid level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable DHCP thresholds or "false" to disable them.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcp_thresholds
my $enable_dhcp_thresholds = $memberdhcp->enable_dhcp_thresholds();
#Modify enable_dhcp_thresholds
$memberdhcp->enable_dhcp_thresholds("true");
```

enable_email_warnings( )

Use this method to set or retrieve the enable_email_warnings value of a member DHCP server. If we enable it, the Infoblox appliance sends an e-mail notification to an administrator if DHCP address usage crosses a high or low watermark threshold.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set enable_email_warnings flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_email_warnings
my $enable_email_warnings = $memberdhcp->enable_email_warnings();
#Modify enable_email_warnings
$memberdhcp->enable_email_warnings("true");
```

enable_gss_tsig( )

Use this method to set or retrieve the member-level enable_gss_tsig flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This is for DHCPv4 GSS-TSIG authentication.

Parameter

Specify "true" to enable GSS-TSIG authentication for DHCP IPv4 or "false" to disable it. The default value is undefined. Setting this flag overrides the enable_gss_tsig attribute of the grid.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_gss_tsig
my $enable_gss_tsig = $memberdhcp->enable_gss_tsig();
#Modify enable_gss_tsig
$memberdhcp->enable_gss_tsig("true");
```

enable_ifmap( )

Use this method to enable or disable the IF-MAP DHCP client feature on this member. Enabling this feature requires a restart of DHCP services.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Boolean to enable or disable the IF-MAP client. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get IF-MAP enable value
my $ifmap_enabled = $memberdhcp->enable_ifmap();
#Enable IF-MAP Client feature
$memberdhcp->enable_ifmap("true");
```

enable_snmp_warnings( )

Use this method to set or retrieve the enable_snmp_warnings value of member DHCP server. When DHCP address usage crosses a watermark threshold, the appliance sends an SNMP trap to the trap receiver that you defined for the grid member level.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set enable_snmp_warnings flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_snmp_warnings
my $enable_snmp_warnings = $memberdhcp->enable_snmp_warnings();
#Modify enable_snmp_warnings
$memberdhcp->enable_snmp_warnings("true");
```

extensible_attributes( )

Use this method to retrieve the extensible attributes associated with a Grid Member DHCP object.

Parameter

none

Returns

The method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $memberdhcp->extensible_attributes();
```

gss_tsig_key( )

Use this method to set or retrieve the DHCPv4 GSS-TSIG key. This setting overrides the grid-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Infoblox::Grid::KerberosKey object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get gss_tsig_key
my $gss_tsig_key = $memberdhcp->gss_tsig_key();

#Set gss_tsig_key
my $kerberos = Infoblox::Cluster::KerberosKey->new(
       principal => 'local1/a.localhost@LOCAL',
$memberdhcp->gss_tsig_key($kerberos);
```

high_water_mark( )

Use this method to set or retrieve the high_water_mark value of a member DHCP server. If the percent of allocated addresses exceeds this watermark, the appliance makes a syslog entry and if enabled, sends an e-mail notification.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number with the percentage value. Range is from 1 to 100. The high watermark value must be greater than the low watermark value. Default value is 95.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get high_water_mark
my $high_water_mark = $memberdhcp->high_water_mark();
#Modify high_water_mark
$memberdhcp->high_water_mark(80);
```

ifmap_password( )

Use this method to set the IF-MAP client password used by the DHCP IF-MAP client on this member. The value can only be set or modified, but not retrieved.

Include the specified parameter to set the attribute value.

Parameter

String for the IF-MAP client password. The default value is undef.

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Modify IF-MAP client password
$memberdhcp->ifmap_password("secret");
```

ifmap_username( )

Use this method to set or retrieve the IF-MAP client username used by the DHCP IF-MAP client on this member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

String for the IF-MAP client username. The default value is undef.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get IF-MAP client username
my $username = $memberdhcp->ifmap_username();
#Modify IF-MAP client username
$memberdhcp->ifmap_username("ifmapclient");
```

ignore_dhcp_option_list_request( )

Use this method to set or retrieve the ignore_dhcp_option_list_request flag of a grid member DHCP. If this flag is set to true all available DHCP options will be returned to the client.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the ignore_dhcp_option_list_request flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ignore_dhcp_option_list_request
my $ignore_dhcp_option_list_request = $memberdhcp->ignore_dhcp_option_list_request();
#Modify ignore_dhcp_option_list_request
$memberdhcp->ignore_dhcp_option_list_request("false");
#Do not override ignore_dhcp_option_list_request
$memberdhcp->ignore_dhcp_option_list_request(undef);
```

ipv6_ddns_enable_option_fqdn( )

Use this method to set or retrieve the ipv6_ddns_enable_option_fqdn value of a member DHCP server. This method controls whether the FQDN option sent by the client is to be used, or if the server can automatically generate the FQDN.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_ddns_enable_option_fqdn method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_ipv6_ddns_enable_option_fqdn attribute to "false".

Note that when ipv6_ddns_enable_option_fqdn is set to a defined value and override_ipv6_ddns_enable_option_fqdn is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_ddns_enable_option_fqdn("true"); $object->override_ipv6_ddns_enable_option_fqdn("false"); will set override_ipv6_ddns_enable_option_fqdn to "false", and the sequence $object->override_ipv6_ddns_enable_option_fqdn("false"); $object->ipv6_ddns_enable_option_fqdn("true"); will result in override_ipv6_ddns_enable_option_fqdn="true".

Parameter

A 32-bit integer (range from 0 to 4294967295) that specifies the number of seconds the IP address for the name is cached. Default value is half of lease time.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_enable_option_fqdn
my $ipv6_ddns_enable_option_fqdn = $memberdhcp->ipv6_ddns_enable_option_fqdn();
#Modify ipv6_ddns_enable_option_fqdn
$memberdhcp->ipv6_ddns_enable_option_fqdn("true");
```

ipv6_ddns_server_always_updates( )

Use this method to set or retrieve the ipv6_ddns_server_always_updates flag. Set this flag to "false" to update DNS only if requested by the client.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_server_always_updates
my $ipv6_ddns_server_always_updates = $memberdhcp->ipv6_ddns_server_always_updates();
#Modify ipv6_ddns_server_always_updates
$memberdhcp->ipv6_ddns_server_always_updates("true");
```

ipv6_ddns_domainname( )

Use this method to set or retrieve the member IPv6 DDNS domain name value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_ddns_domainname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_ddns_domainname attribute to "false".

Note that when ipv6_ddns_domainname is set to a defined value and override_dipv6_ddns_domainname is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_ddns_domainname("domain.com"); $object->override_ipv6_ddns_domainname("false"); will set override_ipv6_ddns_domainname to "false", and the sequence $object->override_ipv6_ddns_domainname("false"); $object->ipv6_ddns_domainname("domain.com"); will result in override_ipv6_ddns_domainname="true".

Parameter

The DDNS domain name in FQDN (Fully Qualified Domain Name) format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_domainname
my $ipv6_ddns_domainname = $memberdhcp->ipv6_ddns_domainname();
#Modify ipv6_ddns_domainname
$memberdhcp->ipv6_ddns_domainname("domain.com");
```

ipv6_ddns_hostname( )

Use this method to set or retrieve the member IPv6 DDNS hostname value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_ddns_hostname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_ddns_hostname attribute to "false".

Note that when ipv6_ddns_hostname is set to a defined value and override_dipv6_ddns_hostname is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_ddns_hostname("host.com"); $object->override_ipv6_ddns_hostname("false"); will set override_ipv6_ddns_hostname to "false", and the sequence $object->override_ipv6_ddns_hostname("false"); $object->ipv6_ddns_hostname("host.com"); will result in override_ipv6_ddns_hostname="true".

Parameter

The DDNS hostname in FQDN (Fully Qualified Domain Name) format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_hostname
my $ipv6_ddns_hostname = $memberdhcp->ipv6_ddns_hostname();
#Modify ipv6_ddns_hostname
$memberdhcp->ipv6_ddns_hostname("host.com");
```

ipv6_ddns_ttl( )

Use this method to set or retrieve the member IPv6 DDNS TTL value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_ddns_ttl method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_ddns_ttl attribute to "false".

Note that when ipv6_ddns_ttl is set to a defined value and override_dipv6_ddns_ttl is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_ddns_ttl(12345); $object->override_ipv6_ddns_ttl("false"); will set override_ipv6_ddns_ttl to "false", and the sequence $object->override_ipv6_ddns_ttl("false"); $object->ipv6_ddns_ttl(12345); will result in override_ipv6_ddns_ttl="true".

Parameter

The preferred DDNS TTL value in seconds. The default is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_ddns_ttl
my $ipv6_ddns_ttl = $memberdhcp->ipv6_ddns_ttl();
#Modify ipv6_ddns_ttl
$memberdhcp->ipv6_ddns_ttl(12345);
```

ipv6_domain_name( )

Use this method to set or retrieve the ipv6_domain_name value of a Grid Member DHCP object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_domain_name method to "true". Setting the parameter to undefined causes the appliance to use the upper level default and automatically resets the override_ipv6_domain_name attribute to "false".

Note that when ipv6_domain_name is set to a defined value and override_ipv6_domain_name is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_domain_name("testdomain"); $object->override_ipv6_domain_name("false"); will set override_ipv6_domain_name to "false", and the sequence $object->override_ipv6_domain_name("false"); $object->ipv6_domain_name("testdomain"); will result in override_ipv6_domain_name="true".

Parameter

The name of the client's domain for resolving names in DNS in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_domain_name
my $ipv6_domain_name = $memberdhcp->ipv6_domain_name();
#Modify ipv6_domain_name
$memberdhcp->ipv6_domain_name("test_domain.com");
```

ipv6_domain_name_servers( )

Use this method to set or retrieve the ipv6_domain_name_servers value of a Grid Member DHCP object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_domain_name_servers method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_ipv6_domain_name_servers attribute to "false".

Note that when ipv6_domain_name_servers is set to a defined value and override_ipv6_domain_name_servers is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_domain_name_servers("2001::1,2001::2"); $object->override_ipv6_domain_name_servers("false"); will set override_ipv6_domain_name_servers to "false", and the sequence $object->override_ipv6_domain_name_servers("false"); $object->ipv6_domain_name_servers("2001::1,2001::2"); will result in override_ipv6_domain_name_servers="true".

Parameter

The domain name server addresses in IPv6 address format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_domain_name_servers
my $ipv6_domain_name_servers = $memberdhcp->ipv6_domain_name_servers();
#Modify ipv6_domain_name_servers
$memberdhcp->ipv6_domain_name_servers("2001::1,2001::2");
```

ipv6_enable_ddns( )

Use this method to set or retrieve the ipv6_enable_ddns flag. Set this flag to enable DHCPv6 dynamic updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_enable_ddns method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_enable_ddns attribute to "false".

Note that when ipv6_enable_ddns is set to a defined value and override_dipv6_enable_ddns is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_enable_ddns("true"); $object->override_ipv6_enable_ddns("false"); will set override_ipv6_enable_ddns to "false", and the sequence $object->override_ipv6_enable_ddns("false"); $object->ipv6_enable_ddns("true"); will result in override_ipv6_enable_ddns="true".

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_enable_ddns
my $ipv6_enable_ddns = $memberdhcp->ipv6_enable_ddns();
#Modify ipv6_enable_ddns
$memberdhcp->ipv6_enable_ddns("true");
```

ipv6_enable_gss_tsig( )

Use this method to set or retrieve the member-level ipv6_enable_gss_tsig flag.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This is for DHCPv6 GSS-TSIG authentication.

Setting this method to a defined value implicitly sets the override_ipv6_enable_gss_tsig method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_ipv6_enable_gss_tsig attribute to "false".

Note that when ipv6_enable_gss_tsig is set to a defined value and override_ipv6_enable_gss_tsig is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_enable_gss_tsig("true"); $object->override_ipv6_enable_gss_tsig("false"); will set override_ipv6_enable_gss_tsig to "false", and the sequence $object->override_ipv6_enable_gss_tsig("false"); $object->ipv6_enable_gss_tsig("true"); will result in override_ipv6_enable_gss_tsig="true".

Parameter

Specify "true" to enable GSS-TSIG authentication for DHCP IPv6 or "false" to disable it. The default value is undefined. Setting this flag overrides the ipv6_enable_gss_tsig attribute of the grid.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_enable_gss_tsig
my $ipv6_enable_gss_tsig = $memberdhcp->ipv6_enable_gss_tsig();
#Modify ipv6_enable_gss_tsig
$memberdhcp->ipv6_enable_gss_tsig("true");
```

ipv6_enable_retry_updates( )

Use this method to set or retrieve the ipv6_enable_retry_updates flag. If this flag is set to "true", the DHCPv6 server retries failed dynamic DNS updates.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_enable_retry_updates method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_enable_retry_updates attribute to "false".

Note that when ipv6_enable_retry_updates is set to a defined value and override_dipv6_enable_retry_updates is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_enable_retry_updates("true"); $object->override_ipv6_enable_retry_updates("false"); will set override_ipv6_enable_retry_updates to "false", and the sequence $object-

>override_ipv6_enable_retry_updates("false"); $object->ipv6_enable_retry_updates("true"); will result in override_ipv6_enable_retry_updates="true".

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_enable_retry_updates
my $ipv6_enable_retry_updates = $memberdhcp->ipv6_enable_retry_updates();
#Modify ipv6_enable_retry_updates
$memberdhcp->ipv6_enable_retry_updates("true");
```

ipv6_generate_hostname( )

Use this method to set or retrieve the ipv6_generate_hostname flag. Set this flag to generate the hostname if it is not sent by the client.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_generate_hostname method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_generate_hostname attribute to "false".

Note that when ipv6_generate_hostname is set to a defined value and override_dipv6_generate_hostname is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_generate_hostname("true"); $object->override_ipv6_generate_hostname("false"); will set override_ipv6_generate_hostname to "false", and the sequence $object->override_ipv6_generate_hostname("false"); $object->ipv6_generate_hostname("true"); will result in override_ipv6_generate_hostname="true".

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_generate_hostname
my $ipv6_generate_hostname = $memberdhcp->ipv6_generate_hostname();
#Modify ipv6_generate_hostname
$memberdhcp->ipv6_generate_hostname("true");
```

ipv6_gss_tsig_key( )

Use this method to set or retrieve the DHCPv6 GSS-TSIG key. This setting overrides the grid-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Infoblox::Grid::KerberosKey object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_gss_tsig_key
my $ipv6_gss_tsig_key = $memberdhcp->ipv6_gss_tsig_key();

#Set ipv6_gss_tsig_key
my $kerberos = Infoblox::Cluster::KerberosKey->new(
       principal => 'local1/a.localhost@LOCAL',
$memberdhcp->ipv6_gss_tsig_key($kerberos);
```

ipv6_kdc_server( )

Use this method to specify or retrieve the IPv4 address or FQDN of the Kerberos server for DHCPv6 GSS-TSIG authentication. This setting overrides the grid-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The Kerberos server IPv4 address or hostname, with a maximum of 1023 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_kdc_server
my $ipv6_kdc_server = $memberdhcp->ipv6_kdc_server();
#Modify ipv6_kdc_server
$memberdhcp->ipv6_kdc_server("10.32.0.8");
```

ipv6_microsoft_code_page( )

Use this method to set or retrieve the Microsoft client DHCP IPv6 code page value of a grid member. This value is the hostname translation code page for Microsoft DHCP IPv6 clients and overrides the grid-level Microsoft DHCP IPv6 client code page.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_microsoft_code_page method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_microsoft_code_page attribute to "false".

Note that when ipv6_microsoft_code_page is set to a defined value and override_dipv6_microsoft_code_page is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_microsoft_code_page("Cyrillic (ISO-5589-5)"); $object->override_ipv6_microsoft_code_page("false"); will set override_ipv6_microsoft_code_page to "false", and the sequence $object->override_ipv6_microsoft_code_page("false"); $object->ipv6_microsoft_code_page("Cyrillic (ISO-5589-5)"); will result in override_ipv6_microsoft_code_page="true".

Parameter

Microsoft DHCP client code page in string format. Valid values are:

None

Arabic (1256)

Arabic (ISO-8859-6)

Baltic (1257)

Baltic (775)

Baltic (ISO-8859-4)

Central Europe (1250)

Cyrillic (1251)

Cyrillic (855)

Cyrillic (ISO-8859-5)

Greek (1253)

Greek (737)

Greek (ISO-8859-7)

Hebrew (1255)

Hebrew (862)

Hebrew (ISO-8859-8)

Japanese Shift-JIS (932)

Korean (949)

Latin 1 (ISO-8859-1)

Latin 2 (ISO-8859-2)

Latin 3 (ISO-8859-3)

Latin 9 (ISO-8859-15)

Latin I (1252)

Latin II (852)

Multilingual Latin I (850)

Russian (866)

Simplified Chinese GBK (936)

Thai (874)

Traditional Chinese Big5 (950)

Turkish (1254)

Turkish (857)

Turkish (ISO-8859-9)

US (437)

Vietnam (1258)

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_microsoft_code_page
my $ipv6_microsoft_code_page = $memberdhcp->ipv6_microsoft_code_page();
#Modify ipv6_microsoft_code_page
$memberdhcp->ipv6_microsoft_code_page("Cyrillic (ISO-8859-5)");
#Do not override ipv6_microsoft_code_page
$memberdhcp->ipv6_microsoft_code_page("None");
```

ipv6_options( )

Use this method to set or retrieve the ipv6 options.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_options method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_options attribute to "false".

Note that when ipv6_options is set to a defined value and override_dipv6_options is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_options([$opt]); $object->override_ipv6_options("false"); will set override_ipv6_options to "false", and the sequence $object->override_ipv6_options("false"); $object->ipv6_options([$opt]); will result in override_ipv6_options="true".

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_options
my $ipv6_options = $memberdhcp->ipv6_options();
#Modify ipv6_options
my $option1 = Infoblox::DHCP::Option->new(
    name   => "dhcp6.rapid-commit",
```

```
    value  => "true"
);
$memberdhcp->ipv6_options([$option1);
```

ipv6_recycle_leases( )

Use this method to set or retrieve the ipv6_recycle_leases flag of the DHCP member.

If the flag is enabled, the leases are kept in recycle bin until one week after expiration. When the flag is disabled, the leases are irrecoverably deleted.

Setting this method to a defined value implicitly sets the override_ipv6_recycle_leases method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_recycle_leases attribute to "false".

Note that when ipv6_recycle_leases is set to a defined value and override_ipv6_recycle_leases is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_recycle_leases("true"); $object->override_ipv6_recycle_leases("false"); will set override_ipv6_recycle_leases to "false", and the sequence $objct->override_ipv6_recycle_leases("false"); $object->ipv6_recycle_leases("true"); will result in override_ipv6_recycle_leases="true".

Parameter

Specify "true" to set the ipv6_recycle_leases flag or "false" to deactivate/unset it. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_recycle_leases
my $ipv6_recycle_leases = $memberdhcp->ipv6_recycle_leases();
#Modify ipv6_recycle_leases
$memberdhcp->ipv6_recycle_leases("false");
```

ipv6_retry_updates_interval( )

Use this method to set or retrieve the ipv6_retry_updates_interval value. Set the retry interval when the member DHCPv6 server makes repeated attempts to send DDNS updates to a DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that represents the retry interval in minutes. The default retry interval is five minutes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_retry_updates_interval
my $ipv6_retry_updates_interval = $memberdhcp->ipv6_retry_updates_interval();
#Modify ipv6_retry_updates_interval
$memberdhcp->ipv6_retry_updates_interval(10);
```

ipv6_server_duid( )

Use this method to set or retrieve the server DHCPv6 unique identifier (DUID) for the grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

DUID of the grid member in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_server_duid
my $ipv6_server_duid = $memberdhcp->ipv6_server_duid();
#Modify ipv6_server_duid
$memberdhcp->ipv6_server_duid("false");
```

ipv6_update_dns_on_lease_renewal( )

Use this method to set or retrieve the ipv6_update_dns_on_lease_renewal flag. This attribute controls whether the DHCP server updates DNS when an IPv6 DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_ipv6_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_ipv6_update_dns_on_lease_renewal attribute to "false".

Note that when ipv6_update_dns_on_lease_renewal is set to a defined value and override_dipv6_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->ipv6_update_dns_on_lease_renewal("true"); $object->override_ipv6_update_dns_on_lease_renewal("false"); will set override_ipv6_update_dns_on_lease_renewal to "false", and the sequence $object->override_ipv6_update_dns_on_lease_renewal("false"); $object->ipv6_update_dns_on_lease_renewal("true"); will result in override_ipv6_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to enable this feature or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6_update_dns_on_lease_renewal
my $ipv6_update_dns_on_lease_renewal = $memberdhcp->ipv6_update_dns_on_lease_renewal();
#Modify ipv6_update_dns_on_lease_renewal
$memberdhcp->ipv6_update_dns_on_lease_renewal("true");
```

kdc_server( )

Use this method to specify or retrieve the IPv4 address or FQDN of the Kerberos server for DHCPv4 GSS-TSIG authentication. This setting overrides the grid-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The Kerberos server IPv4 address or FQDN, with a maximum of 1023 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get kdc_server
my $kdc_server = $memberdhcp->kdc_server();
#Modify  kdc_server
$memberdhcp->kdc_server("10.32.0.7");
```

lease_scavenge_time( )

Use this method to set or retrieve the member-level lease_scavenge_time value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_lease_scavenge_time method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_lease_scavenge_time attribute to "false".

Note that when lease_scavenge_time is set to a defined value and override_lease_scavenge_time is set to "false", the last operation takes precedence. Thus the sequence $object->lease_scavenge_time(604800); $object->override_lease_scavenge_time("false"); will set override_lease_scavenge_time to "false", and the sequence $object->override_lease_scavenge_time("false"); $object->lease_scavenge_time(604800); will result in override_lease_scavenge_time="true".

Parameter

An integer that specifies the period of time, in seconds, that free and backup leases remain in the database before they are automatically deleted. To disable lease scavenging, set the parameter to -1. The minimum positive value must be greater than 86400 seconds (1 day). The default value of this parameter is -1, which means that lease scavenging is disabled.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get lease_scavenge_time
my $lease_scavenge = $memberdhcp->lease_scavenge_time();
#Modify lease_scavenge_time
$memberdhcp->lease_scavenge_time(30 * 24 * 60 * 60);
```

log_lease_events( )

Use this method to set or retrieve the log_lease_events value of a member DHCP server. This value specifies whether the grid member logs lease events.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_log_lease_events method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_log_lease_events attribute to "false".

Note that when log_lease_events is set to a defined value and override_dlog_lease_events is set to "false", the last operation takes precedence. Thus the sequence $object->log_lease_events("true"); $object->override_log_lease_events("false"); will set override_log_lease_events to "false", and the sequence $object->override_log_lease_events("false"); $object->log_lease_events("true"); will result in override_log_lease_events="true".

Parameter

Specify "true" to set log_lease_events flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get log_lease_events
my $log_lease_events = $memberdhcp->log_lease_events();
#Modify log_lease_events
$memberdhcp->log_lease_events("true");
```

low_water_mark( )

Use this method to set or retrieve the low_water_mark value of a member DHCP server. If the percent of allocated addresses drops below this watermark, the appliance makes a syslog entry and if enabled, sends an e-mail notification.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number with the percentage value. Range is from 0 to 100. The low watermark value must be lower than the high watermark value. The default low watermark number is 0.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get low_water_mark
my $low_water_mark = $memberdhcp->low_water_mark();
#Modify low_water_mark
$memberdhcp->low_water_mark(10);
```

name( )

Use this method to retrieve the grid name.

Parameter

None.

Returns

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $memberdns->name();
```

nextserver( )

Use this method to set or retrieve the next server value of a member DHCP server. This value is the IP address or name of the boot file server on which the boot file is stored.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

This option is overridden independently from Infoblox::Grid::Member::DHCP->bootfile() and Infoblox::Grid::Member::DHCP->bootserver().

Parameter

The boot file server IP address or name in FQDN format. The FQDN consists of the host name followed by the domain name (example: abc.com). A boot file server name can have a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nextserver
my $nextserver = $memberdhcp->nextserver();
#Modify nextserver
$memberdhcp->nextserver("1.1.1.1");
#Un-override nextserver
$memberdhcp->nextserver(undef);
```

microsoft_code_page( )

Use this method to set or retrieve the Microsoft client DHCP IPv4 code page value of a grid member. This value is the hostname translation code page for Microsoft DHCP IPv4 clients and overrides the grid-level Microsoft DHCP IPv4 client code page.

Setting microsoft_code_page() to undef removes the override, enabling the appliance to use the grid-level microsoft_code_page setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Microsoft DHCP client code page in string format. Valid values are the same as those for ipv6_microsoft_code_page. See ipv6_microsoft_code_page().

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get microsoft_code_page
my $microsoft_code_page = $memberdhcp->microsoft_code_page();
#Modify microsoft_code_page
$memberdhcp->microsoft_code_page("Cyrillic (ISO-8859-5)");
#Do not override microsoft_code_page
$memberdhcp->microsoft_code_page("None");
```

options( )

Use this method to set or retrieve the options value of a member DHCP server. DHCP options provide network configuration settings and various services available on the network for DHCP member. This setting overrides the grid level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $options = $memberdhcp->options();
#Modify options
my $option1 = Infoblox::DHCP::Option->new(
    name  => "lease-time",
    value => 50
);

my $option2 = Infoblox::DHCP::Option->new(
    name  => "routers",
    value => "1.4.1.1, 2.2.2.2"
);

my $option3 = Infoblox::DHCP::Option->new(
    name  => "nameservers",
    value => "1.1.1.1, 2.2.2.2"
);

my $option4 = Infoblox::DHCP::Option->new(
    name  => "domain-name",
    value => "domainname.com",
);

my $option5 = Infoblox::DHCP::Option->new(
    name  => "broadcast",
    value => "255.0.0.0",
);

$memberdhcp->options([$option1, $option2, $option3, $option4, $option5]);
```

option60_match_rules( )

Use this method to set or retrieve the option60_match_rules value of a member DHCP server. An Option60MatchRule object allows specifying match rules for Option 60 globally. This setting overrides the grid level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Option60MatchRule object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get option60_match_rules
my $options = $memberdhcp->option60_match_rules();
#Modify option60_match_rules
my $option60_match_rule = Infoblox::DHCP::Option60MatchRule->new(
                                  "option_space" => "infospace",
                                  "match_value" => "Nortel-i2004-A",
                                  "is_substring" => "true",
                                  "substring_offset" => "8",
                                  "substring_length" => "14"
                                );

$memberdhcp->option60_match_rules([$option60_match_rule]);
#Do not override option60_match_rules
$memberdhcp->option60_match_rules(undef);
```

override_ddns_ttl( )

The override_ddns_ttl attribute controls whether the ddns_ttl value in the member DHCP server is used, instead of the grid default.

The override_ddns_ttl attribute can be specified explicitly. It is also set implicitly when ddns_ttl is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ddns_ttl. Set the parameter to "false" to inherit the grid-level setting for ddns_ttl.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ddns_ttl
my $override_ddns_ttl=$memberdhcp->override_ddns_ttl( );
#Modifying override_ddns_ttl
$memberdhcp->override_ddns_ttl("true");
```

override_enable_dhcpv6_service( )

The override_enable_dhcpv6_service attribute controls whether the enable_dhcpv6_service value in the DHCP member is used, instead of the grid default.

The override_enable_dhcpv6_service attribute can be specified explicitly. It is also set implicitly when enable_dhcpv6_service is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for enable_dhcpv6_service. Set the parameter to "false" to inherit the grid-level setting for enable_dhcpv6_service.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_enable_dhcpv6_service
my $override_enable_dhcpv6_service=$memberdhcp->override_enable_dhcpv6_service( );
#Modifying override_enable_dhcpv6_service
$memberdhcp->override_enable_dhcpv6_service("true");
```

override_ipv6_ddns_domainname( )

The override_ipv6_ddns_domainname attribute controls whether the ipv6_ddns_domainname value in the DHCP member is used, instead of the grid default.

The override_ipv6_ddns_domainname attribute can be specified explicitly. It is also set implicitly when ipv6_ddns_domainname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_ddns_domainname. Set the parameter to "false" to inherit the grid-level setting for ipv6_ddns_domainname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_ddns_domainname
my $override_ipv6_ddns_domainname=$memberdhcp->override_ipv6_ddns_domainname( );
#Modifying override_ipv6_ddns_domainname
$memberdhcp->override_ipv6_ddns_domainname("true");
```

override_ipv6_ddns_hostname( )

The override_ipv6_ddns_hostname attribute controls whether the ipv6_ddns_hostname value in the DHCP member is used, instead of the grid default.

The override_ipv6_ddns_hostname attribute can be specified explicitly. It is also set implicitly when ipv6_ddns_hostname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_ddns_hostname. Set the parameter to "false" to inherit the grid-level setting for ipv6_ddns_hostname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_ddns_hostname
my $override_ipv6_ddns_hostname=$memberdhcp->override_ipv6_ddns_hostname( );
#Modifying override_ipv6_ddns_hostname
$memberdhcp->override_ipv6_ddns_hostname("true");
```

override_ipv6_ddns_enable_option_fqdn( )

Use this method to set or retrieve the override_ipv6_ddns_enable_option_fqdn value of a Grid Member DHCP object.

The override_ipv6_ddns_enable_option_fqdn attribute can be specified explicitly. It is also set implicitly when ipv6_ddns_enable_option_fqdn is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for ipv6_ddns_enable_option_fqdn. Set the parameter to "false" to inherit the upper-level setting for ipv6_ddns_enable_option_fqdn.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_ipv6_ddns_enable_option_fqdn
my $override_ipv6_ddns_enable_option_fqdn = $memberdhcp->override_ipv6_ddns_enable_option_fqdn();
#Modify override_ipv6_ddns_enable_option_fqdn
$memberdhcp->override_ipv6_ddns_enable_option_fqdn("true");
```

override_ipv6_ddns_ttl( )

The override_ipv6_ddns_ttl attribute controls whether the ipv6_ddns_ttl value in the DHCP member is used, instead of the grid default.

The override_ipv6_ddns_ttl attribute can be specified explicitly. It is also set implicitly when ipv6_ddns_ttl is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_ddns_ttl. Set the parameter to "false" to inherit the grid-level setting for ipv6_ddns_ttl.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_ddns_ttl
my $override_ipv6_ddns_ttl=$memberdhcp->override_ipv6_ddns_ttl( );
#Modifying override_ipv6_ddns_ttl
$memberdhcp->override_ipv6_ddns_ttl("true");
```

override_ipv6_domain_name( )

Use this method to set or retrieve the override_ipv6_domain_name value of a Grid Member DHCP object.

The override_ipv6_domain_name attribute can be specified explicitly. It is also set implicitly when ipv6_domain_name is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for ipv6_domain_name. Set the parameter to "false" to inherit the upper-level setting for ipv6_domain_name.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_ipv6_domain_name
my $override_ipv6_domain_name = $memberdhcp->override_ipv6_domain_name();
#Modify override_ipv6_domain_name
$memberdhcp->override_ipv6_domain_name("true");
```

override_ipv6_domain_name_servers( )

Use this method to set or retrieve the override_ipv6_domain_name_servers value of a Grid Member DHCP object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the upper-level setting for ipv6_domain_name_servers. Set the parameter to "false" to inherit the upper-level setting for ipv6_domain_name_servers.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_ipv6_domain_name_servers
my $override_ipv6_domain_name_servers = $memberdhcp->override_ipv6_domain_name_servers();
#Modify override_ipv6_domain_name_servers
$memberdhcp->override_ipv6_domain_name_servers("true");
```

override_ipv6_enable_ddns( )

The override_ipv6_enable_ddns attribute controls whether the ipv6_enable_ddns value in the DHCP member is used, instead of the grid default.

The override_ipv6_enable_ddns attribute can be specified explicitly. It is also set implicitly when ipv6_enable_ddns is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_enable_ddns. Set the parameter to "false" to inherit the grid-level setting for ipv6_enable_ddns.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_enable_ddns
my $override_ipv6_enable_ddns=$memberdhcp->override_ipv6_enable_ddns( );
#Modifying override_ipv6_enable_ddns
$memberdhcp->override_ipv6_enable_ddns("true");
```

override_ipv6_enable_gss_tsig( )

The override_ipv6_enable_gss_tsig attribute controls whether the ipv6_enable_gss_tsig value in the DHCP member is used, instead of the grid default.

The override_ipv6_enable_gss_tsig attribute can be specified explicitly. It is also set implicitly when ipv6_enable_gss_tsig is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_enable_gss_tsig. Set the parameter to "false" to inherit the grid-level setting for ipv6_enable_gss_tsig.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_enable_gss_tsig
my $override_ipv6_enable_gss_tsig=$memberdhcp->override_ipv6_enable_gss_tsig( );
#Modifying override_ipv6_enable_gss_tsig
$memberdhcp->override_ipv6_enable_gss_tsig("true");
```

override_ipv6_enable_retry_updates( )

The override_ipv6_enable_retry_updates attribute controls whether the ipv6_enable_retry_updates value of the DHCP member is used, instead of the grid default.

The override_ipv6_enable_retry_updates attribute can be specified explicitly. It is also set implicitly when ipv6_enable_retry_updates is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_enable_retry_updates. Set the parameter to "false" to inherit the grid-level setting for ipv6_enable_retry_updates.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_enable_retry_updates
my $override_ipv6_enable_retry_updates=$memberdhcp->override_ipv6_enable_retry_updates( );
#Modifying override_ipv6_enable_retry_updates
```

```
$memberdhcp->override_ipv6_enable_retry_updates("true");
```

override_ipv6_generate_hostname( )

The override_ipv6_generate_hostname attribute controls whether the ipv6_generate_hostname value in the DHCP member is used, instead of the grid default.

The override_ipv6_generate_hostname attribute can be specified explicitly. It is also set implicitly when ipv6_generate_hostname is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_generate_hostname. Set the parameter to "false" to inherit the grid-level setting for ipv6_generate_hostname.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_generate_hostname
my $override_ipv6_generate_hostname=$memberdhcp->override_ipv6_generate_hostname( );
#Modifying override_ipv6_generate_hostname
$memberdhcp->override_ipv6_generate_hostname("true");
```

override_ipv6_microsoft_code_page( )

The override_ipv6_microsoft_code_page attribute controls whether the ipv6_microsoft_code_page value in the DHCP member is used, instead of the grid default.

The override_ipv6_microsoft_code_page attribute can be specified explicitly. It is also set implicitly when ipv6_microsoft_code_page is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_microsoft_code_page. Set the parameter to "false" to inherit the grid-level setting for ipv6_microsoft_code_page.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_microsoft_code_page
my $override_ipv6_microsoft_code_page=$memberdhcp->override_ipv6_microsoft_code_page( );
#Modifying override_ipv6_microsoft_code_page
$memberdhcp->override_ipv6_microsoft_code_page("true");
```

override_ipv6_options( )

The override_ipv6_options attribute controls whether the ipv6_options value in the DHCP member is used, instead of the grid default.

The override_ipv6_options attribute can be specified explicitly. It is also set implicitly when ipv6_options is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_options. Set the parameter to "false" to inherit the grid-level setting for ipv6_options.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_options
my $override_ipv6_options=$memberdhcp->override_ipv6_options( );
#Modifying override_ipv6_options
$memberdhcp->override_ipv6_options("true");
```

override_ipv6_recycle_leases( )

The override_ipv6_recycle_leases attribute controls whether the ipv6_recycle_leases value in the DHCP member is used, instead of the grid default.

The override_ipv6_recycle_leases attribute can be specified explicitly. It is also set implicitly when ipv6_recycle_leases is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_recycle_leases. Set the parameter to "false" to inherit the grid-level setting for ipv6_recycle_leases.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_recycle_leases
my $override_ipv6_recycle_leases=$memberdhcp->override_ipv6_recycle_leases( );
#Modifying override_ipv6_recycle_leases
$memberdhcp->override_ipv6_recycle_leases("true");
```

override_ipv6_update_dns_on_lease_renewal( )

The override_ipv6_update_dns_on_lease_renewal attribute controls whether the ipv6_update_dns_on_lease_renewal value in the DHCP member is used, instead of the grid default.

The override_ipv6_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when ipv6_update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for ipv6_update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for ipv6_update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_ipv6_update_dns_on_lease_renewal
my $override_ipv6_update_dns_on_lease_renewal=$memberdhcp-
>override_ipv6_update_dns_on_lease_renewal( );
#Modifying override_ipv6_update_dns_on_lease_renewal
$memberdhcp->override_ipv6_update_dns_on_lease_renewal("true");
```

override_lease_scavenge_time( )

The override_lease_scavenge_time attribute controls whether the lease_scavenge_time value in the member DHCP server is used, instead of the grid default.

The override_lease_scavenge_time attribute can be specified explicitly. It is also set implicitly when lease_scavenge_time is set to a

defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for lease_scavenge_time. Set the parameter to "false" to inherit the grid-level setting for lease_scavenge_time.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_lease_scavenge_time
my $override_lease_scavenge_time=$memberdhcp->override_lease_scavenge_time( );
#Modifying override_lease_scavenge_time
$memberdhcp->override_lease_scavenge_time("true");
```

override_log_lease_events( )

The override_log_lease_events attribute controls whether the log_lease_events value in the member DHCP server is used, instead of the grid default.

The override_log_lease_events attribute can be specified explicitly. It is also set implicitly when log_lease_events is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for log_lease_events. Set the parameter to "false" to inherit the grid-level setting for log_lease_events.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_log_lease_events
my $override_log_lease_events=$memberdhcp->override_log_lease_events( );
#Modifying override_log_lease_events
$memberdhcp->override_log_lease_events("true");
```

override_preferred_lifetime( )

The override_preferred_lifetime attribute controls whether the preferred_lifetime value in the DHCP member is used, instead of the grid default.

The override_preferred_lifetime attribute can be specified explicitly. It is also set implicitly when preferred_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for preferred_lifetime. Set the parameter to "false" to inherit the grid-level setting for preferred_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_preferred_lifetime
my $override_preferred_lifetime=$memberdhcp->override_preferred_lifetime( );
#Modifying override_preferred_lifetime
$memberdhcp->override_preferred_lifetime("true");
```

override_update_dns_on_lease_renewal( )

The override_update_dns_on_lease_renewal attribute controls whether the update_dns_on_lease_renewal value in the member DHCP server is used, instead of the grid default.

The override_update_dns_on_lease_renewal attribute can be specified explicitly. It is also set implicitly when update_dns_on_lease_renewal is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for update_dns_on_lease_renewal. Set the parameter to "false" to inherit the grid-level setting for update_dns_on_lease_renewal.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_update_dns_on_lease_renewal
my $override_update_dns_on_lease_renewal=$memberdhcp->override_update_dns_on_lease_renewal( );
#Modifying override_update_dns_on_lease_renewal
$memberdhcp->override_update_dns_on_lease_renewal("true");
```

override_valid_lifetime( )

The override_valid_lifetime attribute controls whether the valid_lifetime value in the Grid Member DHCP is used, instead of the upper-level default.

The override_valid_lifetime attribute can be specified explicitly. It is also set implicitly when valid_lifetime is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for valid_lifetime. Set the parameter to "false" to inherit the upper-level setting for valid_lifetime.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_valid_lifetime
my $override_valid_lifetime=$memberdhcp->override_valid_lifetime( );
#Modifying override_valid_lifetime
$memberdhcp->override_valid_lifetime("true");
```

ping_count( )

Use this method to set or retrieve the ping_count value of a member DHCP server. This value specifies the number of pings that the Infoblox appliance sends to an IP address to verify that it is not in use.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Number of pings. Range is from 0 to 10, where 0 disables pings. The default value is 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ping_count
my $ping_count = $memberdhcp->ping_count();
#Modify ping_count
$memberdhcp->ping_count(4);
```

ping_timeout( )

Use this method to set or retrieve the ping_timeout value of a member DHCP server. This value indicates the number of seconds the appliance waits for a response to its ping.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Ping timeout value in seconds. Range is from 1 to 5 seconds. The default value is 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ping_timeout
my $ping_timeout = $memberdhcp->ping_timeout();
#Modify ping_timeout
$memberdhcp->ping_timeout(3);
```

preferred_lifetime( )

Use this method to set or retrieve preferred lifetime value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_preferred_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_preferred_lifetime attribute to "false".

Note that when preferred_lifetime is set to a defined value and override_dpreferred_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->preferred_lifetime(12345); $object->override_preferred_lifetime("false"); will set override_preferred_lifetime to "false", and the sequence $object->override_preferred_lifetime("false"); $object->preferred_lifetime(12345); will result in override_preferred_lifetime="true".

Parameter

Preferred lifetime value in seconds. The default value is 27000.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get preferred_lifetime
my $preferred_lifetime = $memberdhcp->preferred_lifetime();
#Modify preferred_lifetime
$memberdhcp->preferred_lifetime(12345);
```

pxe_lease_time( )

Use this method to set or retrieve the pxe_lease_time value of a member DHCP server. You can specify the duration of time it takes a host to connect to a boot server, such as a TFTP server, and download the file it needs to boot.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer (range from 0 to 4294967295) that specifies the lease time in seconds. Default value is 43200 seconds.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get pxe_lease_time
my $pxe_lease_time = $memberdhcp->pxe_lease_time();
#Modify pxe_lease_time
$memberdhcp->pxe_lease_time(200);
```

recycle_leases( )

Use this method to set or retrieve the recycle_leases value of a member DHCP server. If you enable this check box, and then delete a DHCP range, the appliance stores active leases from this range up to one week after the leases expires.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set recycle_leases flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get recycle_leases
my $recycle_leases = $memberdhcp->recycle_leases();
#Modify recycle_leases
$memberdhcp->recycle_leases("true");
```

retry_ddns_updates( )

Use this method to set or retrieve the retry_ddns_updates value of a member DHCP server. This value indicates whether the DHCP server makes repeated attempts to send DDNS updates to a DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the DHCP server to retry DDNS updates or "false" to disable this feature.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get retry_ddns_updates
my $retry_ddns_updates = $memberdhcp->retry_ddns_updates();
#Modify retry_ddns_updates
$memberdhcp->retry_ddns_updates("true");
```

syslog_facility( )

Use this method to set or retrieve the syslog_facility value of a member DHCP server. The syslog facility is the location on the syslog server to which you want to sort the syslog messages.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are: `"daemon"`, `"local0"`, `"local1"`, `"local2"`, `"local3"`, `"local4"`, `"local5"`, `"local6"`, `"local7"`.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get syslog_facility
my $syslog_facility = $memberdhcp->syslog_facility();
```

```
#Modify syslog_facility
$memberdhcp->syslog_facility("daemon");
```

update_dns_on_lease_renewal( )

The update_dns_on_lease_renewal attribute controls whether the DHCP server updates DNS when a DHCP lease is renewed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_update_dns_on_lease_renewal method to "true". Setting the parameter to undefined causes the appliance to use the grid default and automatically resets the override_update_dns_on_lease_renewal attribute to "false".

Note that when update_dns_on_lease_renewal is set to a defined value and override_update_dns_on_lease_renewal is set to "false", the last operation takes precedence. Thus the sequence $object->update_dns_on_lease_renewal("true"); $object->override_update_dns_on_lease_renewal("false"); will set override_update_dns_on_lease_renewal to "false", and the sequence $object->override_update_dns_on_lease_renewal("false"); $object->update_dns_on_lease_renewal("true"); will result in override_update_dns_on_lease_renewal="true".

Parameter

Specify "true" to set the update_dns_on_lease_renewal flag or "false" to deactivate/unset it.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_dns_on_lease_renewal
my $update_dns_on_lease_renewal=$memberdhcp->update_dns_on_lease_renewal( );
#Modify update_dns_on_lease_renewal
$memberdhcp->update_dns_on_lease_renewal("true");
```

valid_lifetime( )

Use this method to set or retrieve the value of valid_lifetime for Grid Member DHCP.

Setting this method to a defined value implicitly sets the override_valid_lifetime method to "true". Setting the parameter to undefined causes the appliance to use the upper-level default and automatically resets the override_valid_lifetime attribute to "false".

Note that when valid_lifetime is set to a defined value and override_valid_lifetime is set to "false", the last operation takes precedence. Thus the sequence $object->valid_lifetime(20000); $object->override_valid_lifetime("false"); will set override_valid_lifetime to "false", and the sequence $object->override_valid_lifetime("false"); $object->valid_lifetime(20000); will result in override_valid_lifetime="true".

Parameter

Valid lifetime value in seconds. The default value is 43200.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get valid_lifetime
my $valid_lifetime = $memberdhcp->valid_lifetime();
#Modify valid_lifetime
$memberdhcp->valid_lifetime(12345);
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to a member DHCP object, such as get and modify an object. This sample code also includes error handling for the operations.

#Preparation prior to operations on a member DHCP object

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
```

```
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Get and modify the member DHCP property object.

```
#Get the member DHCP object. Considering the id grid member named infoblox.localdomain was previously
created.
my @object = $session->get(
    object => "Infoblox::Grid::Member::DHCP",
    name   => "infoblox.localdomain"
    )
    or die("get member DHCP failed: ",
         $session->status_code(), $session->status_detail());

#Modify member DHCP object.
my $option1 = Infoblox::DHCP::Option->new(
    name  => "lease-time",
    value => 50
);

my $option2 = Infoblox::DHCP::Option->new(
    name  => "domain-name",
    value => "domainname.com",
);

unless (scalar(@object) == 0) {
    my $object = $object[0];

    if ($object) {
        $object->enable_dhcp("true");
        $object->authority("true");
        $object->enable_ddns("true");
        $object->ddns_update_fixed_addresses("true");
        $object->ddns_use_option81("true");
        $object->ddns_server_always_updates("true");
        $object->ddns_generate_hostname("true");
        $object->pxe_lease_time(1000);
        $object->options([$option1, $option2]);
        $object->microsoft_code_page("Cyrillic (ISO-5589-5)");

        #Un-override bootfile
        $object->bootfile(undef);

        #Un-override bootserver
        $object->bootserver(undef);

        #Un-override nextserver
        $object->nextserver(undef);

        #Applying the changes
        $session->modify($object)
            or die("Modify member DHCP failed: ",
                 $session->status_code(), $session->status_detail());

    }
     } else {
    print "No grid member DHCP found.";
     }

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member::DHCP, Infoblox::DHCP::Option, Infoblox::Grid::KerberosKey, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::DHCP::Filter::MAC

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::DNS - Grid Member DNS object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - additional_ip_list( )
  - allow_gss_tsig_zone_updates( )
  - allow_query( )
  - allow_recursive_query( )
  - allow_transfer( )
  - allow_update( )
  - auto_create_a_and_ptr_for_lan2( )
  - auto_sort_views( )
  - blackhole_list( )
  - blacklist_action( )
  - blacklist_log_query( )
  - blacklist_redirect_addresses( )
  - blacklist_redirect_ttl( )
  - blacklist_rulesets( )
  - bind_check_names_policy( )
  - custom_root_name_servers( )
  - dns64_groups( )
  - dnssec_enabled( )
  - dnssec_expired_signatures_enabled( )
  - dnssec_validation_enabled( )
  - dnssec_trusted_keys( )
  - dns_notify_transfer_source_interface( )
  - dns_query_source_interface( )
  - enable_blackhole( )
  - enable_blacklist( )
  - enable_dns( )
  - enable_dns64( )
  - enable_gss_tsig( )
  - extensible_attributes( )
  - facility( )
  - forwarders( )
  - forward_only( )
  - forward_updates( )
  - glue_record_addresses( )
  - gss_tsig_principal_name( )
  - gss_tsig_version_number( )
  - host_name_restriction_policy( )
  - logging_categories( )
  - minimal_resp( )
  - name( )
  - notify_delay( )
  - notify_source_port( )
  - override_dns64( )
  - override_nxdomain_redirect( )
  - nxdomain_redirect( )
  - nxdomain_redirect_addresses( )
  - nxdomain_redirect_ttl( )
  - nxdomain_log_query( )
  - nxdomain_rulesets( )
  - override_blacklist( )
  - override_dnssec( )
  - query_source_port( )
  - recursive_client_limit( )
  - recursive_query_list( )
  - sortlist( )
  - transfer_excluded_servers( )
  - transfer_format( )
  - use_lan2_port( )
  - use_mgmt_port( )
  - use_root_name_servers( )
  - views( )
- SAMPLE CODE

AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::DNS - Grid Member DNS object.

DESCRIPTION

You can configure the DNS properties of a grid member, including enabling/disabling DNS services and other DNS service parameters. The service configurations of a grid are inherited by all members. For this reason, it is recommended that you configure services at the grid level before configuring member services. When you configure DNS services at the grid level, all grid members inherit the grid-level configuration settings, unless you specifically override them for selected members.

CONSTRUCTOR

The Infoblox::Grid::Member::DNS object is automatically generated upon the successful installation of the Infoblox appliance. It does not require manual construction.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a grid member DNS object.

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

Apply the following attributes to get a specific grid member DNS object:

name - Required. Host name in string format extensible_attributes - Optional. A hash reference containing extensible attributes.

Example

```
 my $object   = $session->get(
    object    => "Infoblox::Grid::Member::DNS",
    name      => "infoblox.localdomain",
     );

# get all Grid::Member::DNS objects with a given extensible attribute
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member::DNS",
    extensible_attributes => { 'Site' => 'Santa Clara' });
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use this method to modify the enable_dns  property.
$object->enable_dns("false");
#Submit modification
my $response = $session->modify( $object );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of the DNS properties of a grid member.

additional_ip_list( )

Use this method to add or retrieve a list of additional IP addresses on which DNS is enabled for a grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an IPv4 address added to the IP configuration list of a grid member.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get additional_ip_list
$memberdns->additional_ip_list();
#Modify additional_ip_list
$memberdns->additional_ip_list( ["10.0.0.10"] );
```

allow_gss_tsig_zone_updates( )

Use this method to enable/disable GSS-TSIG zone updates for this member. This setting overrides the grid GSS-TSIG zone update settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to allow GSS-TSIG zone updates for this member or "false" to disallow them. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get allow_gss_tsig_zone_updates
my $allow_gss_tsig_zone_updates = $memberdns->allow_gss_tsig_zone_updates( );
#Modify allow_gss_tsig_zone_updates
$memberdns->allow_gss_tsig_zone_updates("true");
```

allow_query( )

Use this method to add or retrieve an allow_query list.

Select "allow query" to permit queries from the specified IPv4 or IPv6 addresses and networks. The list specifies IP `address(es)` and/or `network(s)` from which queries are allowed or denied. The appliance can also use Transaction Signature (TSIG) keys to authenticate the queries. This setting overrides the grid query settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 or IPv6 addresses, networks and/or Infoblox::DNS::TSIGKey object(s). To deny queries, add "!" as a prefix to `address(es)` and network(s). Specify "any" to allow queries from any address. By default, queries are allowed from any address. Undefine the value to use the Grid::DNS level allow_query list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get allow_query
$memberdns->allow_query();
#Modify allow_query
$memberdns->allow_query( ["any"] );

#Specifying TSIG keys
my $tsig_key1 = $session->gen_tsig_key();
my $query_key = Infoblox::DNS::TSIGKey->new(
  name => "query_key1",
  key  => $tsig_key1
);
#Modify allow_query
$memberdns->allow_query( [ "10.0.0.10" , "20.0.0.0/24"
,"2001::123a","2003::222/64","!"."30.0.0.30" , "!"."40.10.0.0/24","!"."2004::125b","!"."2006::34bc"]
);

#Modify allow_query to use Grid level settings
$memberdns->allow_query(undef);
```

allow_recursive_query( )

Use this method to enable/disable responses to recursive queries. This setting overrides grid recursive query settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to allow responses to recursive queries or "false" to reject recursive queries. The default value is "false". Undefine to use the Grid::DNS level allow_recursive_query setting. Note: You should also undefine the recursive_query_list list to use the Grid::DNS level settings.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get allow_recursive_query
my $allow_recursive_query = $memberdns->allow_recursive_query( );
#Modify allow_recursive_query
$memberdns->allow_recursive_query("true");
```

allow_transfer( )

Use this method to allow or reject zone transfers from specified IPv4 or IPv6 addresses and networks or to allow transfers from hosts authenticated by Transaction signature (TSIG) key. This setting overrides the grid zone transfer settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 or IPv6 addresses, networks and/or Infoblox::DNS::TSIGKey object(s). To reject zone transfers, add '!' as a prefix to IP addresses and networks. Specify "any" to allow any transfer, or undefine to use the Grid::DNS level allow_transfer list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting allow_transfer
my $memberdns = $object->allow_transfer( );
#Modifying allow_transfer
$memberdns->allow_transfer(["any"]);

#Specifying TSIG keys
my $tsig_key1 = $session->gen_tsig_key();
my $key1 = Infoblox::DNS::TSIGKey->new(
  name => "key1",
  key  => $tsig_key1
);
my $key2 = Infoblox::DNS::TSIGKey->new(
  name => "tsig_xfer",
  key  => ":2xCOMPAT"
);
#Modifying allow_transfer
$memberdns->allow_transfer(["10.0.0.10", $key1, "20.0.0.0/24", $key2,
"2004::325f","2007::876c/64","!"."30.0.0.30", "!"."40.10.0.0/24"]);

#Use Grid::DNS level allow_transfer list
$memberdns->allow_transfer(undef);
```

allow_update( )

Use this method to allow or reject dynamic updates from specified IPv4 or IPv6 addresses, networks or from host authenticated by TSIG key. This setting overrides grid update settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 or IPv6 addresses, networks and/or Infoblox::DNS::TSIGKey object(s). To reject updates, add '!' as a prefix to IP addresses and networks. Specify "any" to allow any update, or undefine to use the Grid::DNS level allow_update list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting allow_update
$memberdns->allow_update();
#Modifying allow_update
$memberdns->allow_update(["any"]);

#Creating TSIG Key object.
my $update_key1 = $session->gen_tsig_key();
my $up_key1 = Infoblox::DNS::TSIGKey->new(
  name => "update_key",
  key  => $update_key1
  );
#Modifying allow_update
$memberdns->allow_update([ $up_key1, "10.0.0.10", "20.0.0.0/24","2008::129b","!"."30.0.0.30",
"!"."40.10.0.0/24","!"."2004::76bc/64"]);

#Use Grid::DNS level allow_update list
$memberdns->allow_update(undef);
$memberdns->update_keys(undef);
```

auto_create_a_and_ptr_for_lan2( )

Use this method to enable/disable the auto-generation of A and PTR records for the LAN2 IP address, if DNS service is enabled on LAN2.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the auto-generation of A and PTR records for the LAN2 IP address or "false" to disable it. Default is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting auto_create_a_and_ptr_for_lan2
my $auto_create_a_and_ptr_for_lan2 = $memberdns->auto_create_a_and_ptr_for_lan2( );
#Modifying auto_create_a_and_ptr_for_lan2
$memberdns->auto_create_a_and_ptr_for_lan2("true");
```

auto_sort_views( )

Use this method to enable/disable a grid member to automatically sort DNS views. The order of the DNS views determines the order in which the appliance checks the match lists.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the automatic sorting of DNS views or "false" to disable it. Default is "false"

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting auto_sort_views
$memberdns->auto_sort_views( );
#Modifying auto_sort_views
$memberdns->auto_sort_views("true");
```

blackhole_list( )

Use this method to set or retrieve the list of IPv4 or IPv6 addresses and networks from which DNS queries are blocked. This setting overrides the grid blackhole_list.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A valid value is an array reference that contains IPv4 or IPv6 addresses and networks. To allow queries, add '!' as the prefix to IP addresses and networks. Specify "any" to block queries or undefine the usage of Grid::DNS level blackhole_list. Note: You should also undefine the enable_blackhole list to use Grid::DNS level settings.

Returns

When you specify a parameter and the modification succeeds, the method returns "true". When the operation fails, the method returns "false".

When you do not specify a parameter, the method returns the attribute value.

Example

```
    #Getting blackhole_list
    my $blackhole_list = $memberdns->blackhole_list( );
    #Modifying blackhole_list
    $memberdns->blackhole_list(["any"]);
    #Modifying blackhole_list
    $memberdns->blackhole_list(["10.0.0.10",
"20.0.0.0/24","2010::256/64","!"."2011::23cb/64","!"."30.0.0.30", "!"."40.10.0.0/24"]);
    #Use Grid::DNS level blackhole_list
    $memberdns->blackhole_list(undef);
    $memberdns->enable_blackhole(undef);
```

blacklist_action( )

Use this method to specify the action to perform when a domain name matches the pattern defined in a rule that is specified by the blacklist_ruleset method.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "REDIRECT" or "REFUSE". The default value is "REFUSE".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Examples

```
  #Get blacklist_action
  my $blacklist_action = $memberdns->blacklist_action();
  #Modify blacklist_action
  $memberdns->blacklist_action("REDIRECT");
```

blacklist_log_query( )

Use this method to set or retrieve the flag that indicates whether blacklist redirection queries are logged.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable logging, or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
   #Get blacklist_log_query
   my $blacklist_log_query = $memberdns->nxdomain_log_query( );
   #Modify blacklist_log_query
   $memberdns->blacklist_log_query("true");
```

blacklist_redirect_addresses( )

Use this method to specify the IP addresses the appliance includes in the response it sends in place of a blacklisted IP address.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array of IPv4 addresses. The default value is undefined.

Setting this parameter to ["10.0.0.10", "10.0.0.20"] or [], implicitly sets the override_blacklist_redirect attribute to "true". Setting this parameter to undefined causes the appliance to use the grid-level blacklist redirection settings and automatically sets the override_blacklist_redirect attribute to "false" if "enable_blacklist_redirect" is also undefined. But if "enable_blacklist_redirect" contains a value, then the override_blacklist_redirect attribute is set to "true".

Note that when blacklist_redirect_addresses contains a valid array reference and override_blacklist_redirect is set to "false", the last operation takes precedence. Thus the sequence $memberdns->blacklist_redirect_addresses(["10.0.0.10", "10.0.0.20"]); $memberdns->override_blacklist_redirect("false"); sets override_blacklist_redirect to "false", and the sequence $memberdns->override_blacklist_redirect("false"); $memberdns->blacklist_redirect_addresses(["10.0.0.10", "10.0.0.20"]); sets override_blacklist_redirect to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_redirect_addresses
my $blacklist_redirect_addresses = $memberdns->blacklist_redirect_addresses( );
#Modify blacklist_redirect_addresses
$memberdns->blacklist_redirect_addresses(["10.0.0.10", "10.0.0.20"]);
```

blacklist_redirect_ttl( )

Use this method to set or retrieve the TTL value of the synthetic DNS responses that result from blacklist redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer, from 0 to 4294967295, that represents the TTL in seconds. The default value is 60.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_redirect_ttl
my $blacklist_redirect_ttl = $memberdns->blacklist_redirect_ttl( );
#Modify blacklist_redirect_ttl
$memberdns->blacklist_redirect_ttl(60);
```

blacklist_rulesets( )

Use this method to specify or retrieve Infoblox::DNS::Ruleset object names assigned at the grid level for blacklist redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains the names of the Infoblox::DNS::Ruleset objects assigned to the grid. The names must be in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get blacklist_rulesets
my $blacklist_rulesets = $memberdns->blacklist_rulesets();
#Modify blacklist_rulesets
$memberdns->blacklist_rulesets(["ruleset1", "ruleset2"]);
```

bind_check_names_policy( )

Use this method to set or retrieve the BIND check names policy, which indicates the action the appliance takes when it encounters host names that do not comply with the Strict Hostname Checking policy. This method applies only if the host name restriction policy is set to "Strict Hostname Checking".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The allowed values of the parameter are "fail", "warn" or undefined. This indicates the desired BIND behavior in the case of illegal hostnames. The default value is undefined, which means the appliance ignores illegal hostnames.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get bind_check_names_policy value
my $names_policy = $memberdns->bind_check_names_policy();
#Modify bind_check_names_policy
$memberdns->bind_check_names_policy("warn");
```

custom_root_name_servers( )

Use this method to set or retrieve the list of customized root nameserver(s). You can either select and use Internet root name servers or specify custom root name servers by providing a host name and IP address to which the Infoblox appliance can send queries. Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::DNS::RootNameServer object(s).

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the root nameserver list
my $rootNS_list = $memberdns->custom_root_name_servers();
#Modify the root nameserver list
#Creating RootNameServer objects.
my $rootns1 = Infoblox::DNS::RootNameServer->new(
    ipv4addr  => "4.4.4.4",
    host_name => "rns1.test.com",
    );

my $rootns2 = Infoblox::DNS::RootNameServer->new(
    ipv4addr  => "5.5.5.5",
    host_name => "rns2.test.com",
    );

$memberdns->custom_root_name_servers([$rootns1, $rootns2]);
```

dns64_groups( )

Use this method to set or retrieve the list of DNS64 synthesis groups associated with this member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_dns64 method to "true". Setting the parameter to undefined causes the appliance to use the grid DNS default and automatically resets the override_dns64 method to "false".

Note that when dns64_groups is set to a defined value and override_dns64_groups is set to "false", the last operation takes precedence. Thus the sequence $object->dns64_groups([$group1]); $object->override_dns64("false"); will set override_dns64 to "false", and the sequence $object->override_dns64([$group1]); $object->dns64_groups("testdomain"); will result in override_dns64="true".

Parameter

Valid value is an array reference that contains Infoblox::Grid::DNS::DNS64Group objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get the list of DNS64 synthesis groups
my $dns64_groups_list = $memberdns->dns64_groups();

#Modify it
my $group = $session->get(
                        object => "Infoblox::Grid::DNS::DNS64Group"),
                        name => "groupname",
                        prefix => "groupprefix",
                      );
$memberdns->dns64_groups([$group]);
```

dnssec_enabled( )

Use this method to enable/disable the DNSSEC feature.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to enable the DNSSEC feature. Set the parameter to "false" to disable the feature.

Setting this parameter to "true" or "false" implicitly sets the override_dnssec attribute to "true". Setting the parameter to undefined causes the appliance to use the grid-level DNSSEC settings and automatically sets the override_dnssec attribute to "false".

Note that in the case when dnssec_enabled is set to "true"/"false" and override_dnssec is set to "false" simultaneously, the last operation takes precedence. Thus the sequence $grid_member->dnssec_enabled("false"); $grid_member->override_dnssec("false"); will set override_dnssec to "false", and the sequence $grid_member->override_dnssec("false"); $grid_member->dnssec_enabled("false"); will result in override_dnssec="true".

The default value of the parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_enabled
my $dnssec_enabled = $memberdns->dnssec_enabled();
#Modify dnssec_enabled
$memberdns->dnssec_enabled("true");
#Set the value to undefined to inherit grid level settings.
$memberdns->dnssec_enabled(undef);
```

dnssec_expired_signatures_enabled( )

Use this method to control whether the DNS member accepts expired signatures.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to accept expired signatures. Set it to "false" to reject expired signatures.

The default value of the parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_expired_signatures_enabled
my $dnssec_expired_signatures_enabled = $memberdns->dnssec_expired_signatures_enabled();
#Modify dnssec_expired_signatures_enabled
$memberdns->dnssec_expired_signatures_enabled("true");
```

dnssec_validation_enabled( )

Use this method to enable or disable DNS security validation.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to enable the validation. Set it to "false" to disable the validation.

The default value of the parameter is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dnssec_validation_enabled
my $dnssec_validation_enabled = $memberdns->dnssec_validation_enabled();
#Modify dnssec_validation_enabled
$memberdns->dnssec_validation_enabled("false");
```

dnssec_trusted_keys( )

Use this method to specify the list of trusted keys for the DNSSEC feature.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Array reference that contains a list of Infoblox::DNS::DnssecTrustedKey objects.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns a reference to a list of the trusted keys.

Example

```
#Get dnssec_trusted_keys
my $dnssec_trusted_keys = $memberdns->dnssec_trusted_keys();
#Modify dnssec_trusted_keys
my $key1=Infoblox::DNS::DnssecTrustedKey->new(
   fqdn => "myzone.com",
   algorithm => "RSAMD5",
   key       => "c2ltcGxlIGtleQo="
   );
$memberdns->dnssec_trusted_keys([$key1]);
```

dns_notify_transfer_source_interface( )

Use this method to specify which IP address is used as the source for DDNS notify and transfer operations.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "VIP", "MGMT", "LAN2", "Any" or an IPV4 address. Default is "VIP".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting dns_source_interface
my $dns_source_interface = $memberdns->dns_source_interface( );
#Modifying dns_source_interface
$memberdns->dns_source_interface("VIP");
```

dns_query_source_interface( )

Use this method to specify which IP address is used as the source for DDNS query operations.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "VIP", "MGMT", "LAN2", "Any" or an IPV4 address. Default is "VIP".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting dns_source_interface
my $dns_source_interface = $memberdns->dns_source_interface( );
#Modifying dns_source_interface
$memberdns->dns_source_interface("VIP");
```

enable_blackhole( )

Use this method to enable or disable the blocking of DNS queries. This setting overrides the grid enable_blackhole settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to block DNS queries or "false" to disable blocking DNS queries. The default value is "false".

Returns

When you specify a parameter and the modification succeeds, the method returns "true". When the operation fails, the method returns "false".

When you do not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_blackhole
my $enable_blackhole = $object->enable_blackhole( );
#Modify enable_blackhole
$memberdns->enable_blackhole("true");
```

enable_blacklist( )

Use this method to enable or disable a blacklist on the grid member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_blacklist flag or "false" to deactivate/unset it. The default is "false".

Setting this parameter to "true" or "false" implicitly sets the override_blacklist attribute to "true". Setting the parameter to undefined causes the appliance to use the grid-level blacklist settings and automatically sets the override_blacklist attribute to "false" if "blacklist_redirect_addresses" is also undefined.

Note that when enable_blacklist is set to "true"/"false" and override_blacklist is set to "false", the last operation takes precedence. Thus the sequence $memberdns->enable_blacklist("false"); $memberdns->override_blacklist("false"); sets override_blacklist to "false", and the sequence $memberdns->override_blacklist("false"); $memberdns->enable_blacklist("false"); sets override_blacklist to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_blacklist
my $enable_blacklist = $memberdns->enable_blacklist( );
#Modifying enable_blacklist
$memberdns->enable_blacklist("true");
```

enable_dns( )

Use this method to enable/disable the DNS service of a member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable DNS service or "false" to disable it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_dns
$memberdns->enable_dns( );
#Modifying enable_dns
$memberdns->enable_dns("true");
```

enable_dns64( )

Use this method to enable/disable DNS64 support for this member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting this method to a defined value implicitly sets the override_dns64 method to "true". Setting the parameter to undefined causes the appliance to use the grid DNS default and automatically resets the override_dns64 method to "false".

Note that when enable_dns64 is set to a defined value and override_enable_dns64 is set to "false", the last operation takes precedence. Thus the sequence $object->enable_dns64("true"); $object->override_dns64("false"); will set override_dns64 to "false", and the sequence $object->override_dns64("true"); $object->enable_dns64("testdomain"); will result in override_dns64="true".

Parameter

Specify "true" to set the enable_dns64 flag or "false" to deactivate and unset it. The default value is "false".

Returns

When you specify a parameter and the modification succeeds, the method returns "true". When the operation fails, the method returns "false".

When you do not specify a parameter, the method returns the attribute value.

Example

```
#Get the enable_dns64 flag
my $enable_dns64 = $memberdns->enable_dns64();
#Modify the enable_dns64 flag
$memberdns->enable_dns64("true");
```

enable_gss_tsig( )

Use this method to enable/disable the appliance to receive GSS-TSIG authenticated updates from DHCP clients. You must import the keytab file before you enable this feature. See Infoblox::Session->import_data().

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the the appliance to receive GSS-TSIG authenticated updates or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_gss_tsig
$my $enable_gss_tsig = $memberdns->enable_gss_tsig( );
#Modifying enable_gss_tsig
$session->import_data(
     type        => 'keytab',
     path        => '/infoblox/tests/datasets/gss-tsig-gs.keytab',
     service     => 'dns',
```

```
        virtual_ip  => '192.168.1.2');
$memberdns->enable_gss_tsig("true");
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a Grid Member DNS object.

This method is read-only. Use the Infoblox::Grid::Member object to set or modify the grid member extensible attributes.

Returns

The method returns the attribute value. Attempting to modify the extensible attributes produces an error.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $memberdns->extensible_attributes();
```

facility( )

Use this method to specify a syslog facility. This is the location on the syslog server to which you want to sort the DNS logging messages. This setting overrides the grid logging facility settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Values are: "daemon", "local0", "local1", "local2", "local3", "local4", "local5", "local6", and "local7".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get facility
$memberdns->facility( );
#Modify facility
$memberdns->facility("local0");
```

forwarders( )

Use this method to specify forwarders for the member. A forwarder is essentially a name server to which other name servers first send all of their off-site queries. The forwarder builds up a cache of information, avoiding the need for the other name servers to send queries off-site. This setting overrides the grid-level setting.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::DNS::Member, Infoblox::DNS::Nameserver object(s), or IP addresses.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting forwarders
$memberdns->forwarders( );
#Modifying forwarders
$memberdns->forwarders( [ "10.0.0.10" ] );
```

forward_only( )

Use this method to configure this member to send queries to forwarders only. When the value is "true", the member sends queries to forwarders only, and not to other internal or Internet root servers.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable forward only or "false" to desable it. Default is "false"

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting forward_only
$memberdns->forward_only( );
#Modifying forward_only
$memberdns->forward_only("true");
```

forward_updates( )

Use this method to allow/disallow secondary servers to forward updates to the DNS server. This setting overrides grid update settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable forward updates or "false" desable it. Default is "false"

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting forward_updates
$memberdns->forward_updates( );
#Modifying forward_only
$memberdns->forward_updates("true");
```

glue_record_addresses( )

Use this method to specify which IP address the appliance uses when it generates the glue record. It can use the IP address of the appliance, the NAT IP address, or an IP address that you define.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::DNS::GlueRecordAddr object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $glue_record_addr1 = Infoblox::DNS::GlueRecordAddr->new(
    view               => "default",  #can only be one of existing views
    glue_record_address => "NAT",
);
$memberdns->glue_record_addresses([$glue_record_addr1]);
```

gss_tsig_principal_name( )

Use this method to retrieve the GSS-TSIG principal name from the keytab file you uploaded using Infoblox::Session->import_data().

Returns

The method returns the read-only attribute value.

Example

```
#Getting gss_tsig_principal_name
my $gss_tsig_principal_name = $memberdns->gss_tsig_principal_name();
```

gss_tsig_version_number( )

Use this method to retrieve the GSS-TSIG version number from the keytab file you uploaded using Infoblox::Session->import_data().

Returns

The method returns the read-only attribute value.

Example

```
#Getting gss_tsig_version_number
my $gss_tsig_version_number = $memberdns->gss_tsig_version_number();
```

host_name_restriction_policy( )

Use this method to set or retrieve the member-level host name restriction policy that overrides the grid-level settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is the name of record name policy defined at the grid level via Infoblox::Grid::RecordNamePolicy

There are three predefined policies:

Policy Name Regex Applied

"Allow Any" .+

"Allow Underscore" ^[-a-zA-Z0-9_]+$

"Strict Hostname Checking" ^[a-zA-Z0-9]$|^[a-zA-Z0-9][-a-zA-Z0-9]*[a-zA-Z0-9]$

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting host_name_restriction_policy
$memberdns->host_name_restriction_policy( );
#Modifying host_name_restriction_policy
$memberdns->host_name_restriction_policy("Allow Any");
#Un-override host_name_restriction_policy
$memberdns->host_name_restriction_policy(undef);
```

logging_categories( )

Use this method to set logging categories for this member. This setting overrides grid logging category settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains one or more of the following values:

`"general"`: Records the BIND messages that are not specifically classified.

`"config"`: Records the configuration file parsing messages.

`"dnssec"`: Records the DNSSEC-signed responses.

`"network"`: Records the network operation messages.

`"queries"`: Records the query response messages.

`"security"`: Records the approved and denied requests.

`"xfer-in"`: Records zone transfer messages from the remote name servers to the appliance.

`"xfer-out"`: Records zone transfer messages from the Infoblox appliance to remote name servers.

`"update"`: Records the dynamic update instances.

`"resolver"`: Records the DNS resolution instances, including recursive queries from resolvers.

`"notify"`: Records the asynchronous zone change notification messages.

`"lame_servers"`: Records bad delegation instances.

"database": Records BIND's internal database processes.

"client": Records client requests.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting logging_categories
$memberdns->logging_categories( );
#Modifying logging_categories
$memberdns->logging_categories( [ "general", "database" , "queries" ] );
```

minimal_resp( )

Use this method to enable/disable the ability to return a minimal amount of data in response to a query. This capability speeds up the DNS services provided by the appliance.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable minimal responses or "false" to desable it. Default is "true"

Returns

If you specified a parameter, the method returns 1 when the modification succeeds, and returns other values when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting minimal_resp
$memberdns->minimal_resp( );
#Modifying minimal_resp
$memberdns->minimal_resp("true");
```

name( )

Use this method to retrieve the grid name.

Parameter

None.

Returns

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $memberdns->name();
```

notify_delay( )

Use this method to specify with how many seconds of delay the notify messages are sent to secondaries.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are unsigned integer between 5 and 86400, inclusive. Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting notify_delay
my $notify_delay = $memberdns->notify_delay( );
```

```
#Modifying notify_delay
$memberdns->notify_delay(5);
```

notify_source_port( )

Use this method to specify the source port for notify messages. When requesting zone transfers from the primary server, some secondary DNS servers use the source port number (the primary server used to send the notify message) as the destination port number in the zone transfer request. This setting overrides grid static source port settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Source port number for notify messages. Valid values are between 1 and 63999. The default is picked by BIND.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting notify_source_port
$memberdns->notify_source_port( );
#Modifying notify_source_port
$memberdns->notify_source_port("2");
```

override_dns64( )

The override_dns64 attribute controls whether the DNS64 method values of the member are used, instead of the grid default.

The override_dns64 attribute can be specified explicitly. It is also set implicitly when enable_dns64 or dns64_groups are set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting for enable_dns64 and dns64_groups. Set the parameter to "false" to inherit the grid-level setting for enable_dns64 and dns64_groups.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_dns64
my $override_dns64=$memberdns->override_dns64( );
#Modifying override_dns64
$memberdns->override_dns64("true");
```

override_nxdomain_redirect( )

The "override_nxdomain_redirect" attribute controls whether the "nxdomain_redirect", "nxdomain_redirect_addresses", "nxdomain_redirect_ttl", "nxdomain_log_query", and "nxdomain_rulesets" values for this member are used. Note that these values can be overridden at the DNS view level as well.

The "override_nxdomain_redirect" attribute can be specified explicitly. It is also set implicitly as follows: "nxdomain_redirect" => "true"|"false" or "nxdomain_redirect_addresses" => ["10.0.0.10", "10.0.0.20"]|[] result in "override_nxdomain_redirect" => "true", "nxdomain_redirect" => undef and "nxdomain_redirect_addresses" => undef result in "override_nxdomain_redirect" => "false". If "override_nxdomain_redirect" is specified explicitly as "true", and "nxdomain_redirect" => undef or "nxdomain_redirect_addresses" => undef, then the explicit value takes precedence; and "override_nxdomain_redirect"=>"true", "nxdomain_redirect", and "nxdomain_redirect_addresses" remain unchanged in the database.

Specifying the "nxdomain_redirect_ttl", "nxdomain_log_query", or "nxdomain_rulesets" attributes does not affect the "override_nxdomain_redirect" value. It is implicitly set only by "nxdomain_redirect".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting of the NXDOMAIN redirection attributes. Set the parameter to

"false" to inherit the grid-level settings.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_nxdomain_redirect
my $override_nxdomain_redirect=$memberdns->override_nxdomain_redirect( );
#Modify override_nxdomain_redirect
$memberdns->override_nxdomain_redirect("true");
```

nxdomain_redirect( )

Use this method to enable or disable NXDOMAIN redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the nxdomain_redirect flag or "false" to deactivate/unset it. The default is "false".

Setting this parameter to "true" or "false" implicitly sets the override_nxdomain_redirect attribute to "true". Setting the parameter to undefined causes the appliance to use the grid-level NXDOMAIN redirection settings and automatically sets the override_nxdomain_redirect attribute to "false" if nxdomain_redirect_addresses is also undefined. But if nxdomain_redirect_addresses contains a value, then the override_nxdomain_redirect attribute is set to true.

Note that when nxdomain_redirect is set to "true"/"false" and override_nxdomain_redirect is set to "false", the last operation takes precedence. Thus the sequence $grid_member->nxdomain_redirect("false"); $grid_member->override_nxdomain_redirect("false"); sets override_nxdomain_redirect to "false", and the sequence $grid_member->override_nxdomain_redirect("false"); $grid_member->nxdomain_redirect("false"); sets override_nxdomain_redirect to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting nxdomain_redirect
my $nxdomain_redirect = $grid_member->nxdomain_redirect( );
#Modifying nxdomain_redirect
$grid_member->nxdomain_redirect("true");
```

nxdomain_redirect_addresses( )

Use this method to set the NXDOMAIN redirection addresses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array of IPv4 addresses. The default value is undefined.

Setting this parameter to ["10.0.0.10", "10.0.0.20"] or [] implicitly sets the override_nxdomain_redirect attribute to "true". Setting the parameter to undefined causes the appliance to use the grid-level NXDOMAIN redirection settings and automatically sets the override_nxdomain_redirect attribute to "false" if nxdomain_redirect is also undefined. But if nxdomain_redirect contains a value, then the override_nxdomain_redirect attribute is set to true.

Note that when nxdomain_redirect_addresses contains a valid array reference and override_nxdomain_redirect is set to "false", the last operation takes precedence. Thus the sequence $grid_member->nxdomain_redirect_addresses(["10.0.0.10", "10.0.0.20"]); $grid_member->override_nxdomain_redirect("false"); sets override_nxdomain_redirect to "false", and the sequence $grid_member->override_nxdomain_redirect("false"); $grid_member->nxdomain_redirect_addresses(["10.0.0.10", "10.0.0.20"]); sets override_nxdomain_redirect to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_redirect_addresses
my $nxdomain_redirect_addresses   $grid_member   nxdomain_redirect_addresses
```

```
=              ->                          ( );
#Modify nxdomain_redirect_addresses
$grid_member->nxdomain_redirect_addresses(["10.0.0.10", "10.0.0.20"]);
```

nxdomain_redirect_ttl( )

Use this method to set or retrieve the TTL value of synthetic DNS responses that result from NXDOMAIN redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A 32-bit integer, from 0 to 4294967295, that represents the TTL in seconds. The default value is 60.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_redirect_ttl
my $nxdomain_redirect_ttl = $grid_member->nxdomain_redirect_ttl( );
#Modify nxdomain_redirect_ttl
$grid_member->nxdomain_redirect_ttl(60);
```

nxdomain_log_query( )

Use this method to set or retrieve the flag that indicates whether NXDOMAIN redirection queries are logged.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable logging, or "false" to deactivate it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_log_query
my $nxdomain_log_query = $grid_member->nxdomain_log_query( );
#Modify nxdomain_log_query
$grid_member->nxdomain_log_query("true");
```

nxdomain_rulesets( )

Use this method to specify or retrieve Infoblox::DNS::Ruleset object names assigned at the grid level for NXDOMAIN redirection.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains the names of the Infoblox::DNS::Ruleset objects assigned to the grid. The names must be in string format. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get nxdomain_rulesets
my $nxdomain_rulesets = $grid_member->nxdomain_rulesets();
#Modify nxdomain_rulesets
$grid_member->nxdomain_rulesets(["ruleset1", "ruleset2"]);
```

override_blacklist( )

The "override_blacklist" attribute controls whether the "enable_blacklist", "blacklist_redirect_addresses", "blacklist_redirect_ttl", "blacklist_log_query", and "blacklist_rulesets" values in the database are used.

The "override_blacklist" attribute can be specified explicitly. It is also set implicitly as follows: "enable_blacklist" => "true"|"false" or "blacklist_redirect_addresses" => ["10.0.0.10", "10.0.0.20"]|[] results in "override_blacklist" => "true"; "enable_blacklist" => undef and "blacklist_redirect_addresses" => undef result in "override_blacklist" => "false". If "override_blacklist" is specified explicitly as "true", and "enable_blacklist" => undef or "blacklist_redirect_addresses" => undef, then the explicit value takes precedence. Therefore, "override_blacklist"=>"true", "enable_blacklist"=>undef, and "blacklist_redirect_addresses"=>undef remain unchanged in the database.

Specifying the "blacklist_redirect_ttl", "blacklist_log_query", or "blacklist_rulesets" attributes does not affect the "override_blacklist" value. It is implicitly set only by "enable_blacklist" and "blacklist_redirect_addresses".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level settings of the blacklist redirect attributes. Set the parameter to "false" to inherit the grid-level settings. The default is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get override_blacklist
my $override_blacklist=$memberdns->override_blacklist( );
#Modify override_blacklist
$memberdns->override_blacklist("true");
```

override_dnssec( )

The "override_dnssec" attribute controls whether the "dnssec_enabled", "dnssec_validation_enabled", "dnssec_expired_signatures_enabled", and "dnssec_trusted_keys" values in the database are used. Note that these values can be overridden at the DNS view level as well.

The "override_dnssec" attribute can be specified explicitly. It is also established implicitly as follows: "dnssec_enabled" => "true"|"false" results in "override_dnssec" => "true" "dnssec_enabled" => undef results in "override_dnssec" => "false" If "override_dnssec" is specified explicitly as "true", while "dnssec_enabled"=>undef, the explicit value takes precedence, and "override_dnssec"=>"true" while "dnssec_enabled" remains unchanged in the database.

Supplying the "dnssec_validation_enabled", "dnssec_expired_signatures_enabled", or "dnssec_trusted_keys" attributes does not affect the "override_dnssec" value. It is implicitly affected only by "dnssec_enabled".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the grid-level setting of DNSSEC attributes. Set the parameter to "false" to inherit the grid-level settings.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_dnssec
my $override_dnssec=$memberdns->override_dnssec( );
#Modifying override_dnssec
$memberdns->override_dnssec("true");
```

query_source_port( )

Use this method to specify the source port for queries. Specifying a source port number for recursive queries ensures that a firewall will allow the response. This setting overrides grid static source port settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Source port number for queries. Valid values are between 1 and 63999. The default is picked by BIND.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting query_source_port
$memberdns->query_source_port( );
#Modifying notify_source_port
$memberdns->query_source_port("2");
```

recursive_client_limit( )

Use this method to set a limit on the number of concurrent recursive clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Maximum number of concurrent recursive clients. Range is from 0 to 4294967295

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting recursive_client_limit
$memberdns->recursive_client_limit( );
#Modifying recursive_client_limit
$memberdns->recursive_client_limit("10");
```

recursive_query_list( )

Use this method to set or retrieve the list of IPv4 or IPv6 addresses, networks or hosts authenticated by Transaction signature (TSIG) key from which recursive queries are allowed or denied. This setting overrides grid recursive query list settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 or IPv6 addresses, networks and/or Infoblox::DNS::TSIGKey objects. To deny queries, add '!' as a prefix to IP addresses and networks. Specify "any" to allow any recursive query, or undefine to use the Grid::DNS level recursive_query_list list. Note: You should also undefine the allow_recursive_query list to use Grid::DNS level settings.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting recursive_query_list
$memberdns->recursive_query_list( );
#Modifying recursive_query_list
$memberdns->recursive_query_list(["any"]);

#Creating TSIG key object
my $tsig_key2 = $session->gen_tsig_key();
my $key2 = Infoblox::DNS::TSIGKey->new(
   name => "rcl_key",
   key  => $tsig_key2
);
#Modifying recursive_query_list
$memberdns->recursive_query_list(["10.0.0.10", $key2,
"20.0.0.0/24","2010::256/64","!"."2011::23cb/64","!"."30.0.0.30", "!"."40.10.0.0/24"]);

#Use Grid::DNS level recursive_query_list
$memberdns->recursive_query_list(undef);
$memberdns->allow_recursive_query(undef);
```

sortlist( )

Use this method to create a sort list.

A sort list determines the order of addresses in responses made to DNS queries. This setting overrides grid sort list settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains Infoblox::DNS::Sortlist object(s).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $sort1 = Infoblox::DNS::Sortlist->new (
                        "source_ipv4addr" => "1.2.3.4",
                        "match_list"      => [ "10.20.1.0/24" , "10.20.2.0/24"]
             );
$memberdns_object->sortlist( [ $sort1 ] );
```

transfer_excluded_servers( )

Use this method to exclude specified DNS servers during zone transfers. This setting overrides grid zone transfer settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains IP addresses of the DNS server.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#This means that it will set zone transfer format to Many answers except these servers.
$memberdns->transfer_format("many_answers");
$memberdns->transfer_excluded_servers(["10.0.0.10", "20.0.0.0"]);
```

transfer_format( )

Use this method to set or retrieve the BIND format for a zone transfer. This provides tracking capabilities for single or multiple transfers and their associated servers. This setting overrides grid zone transfer format settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is one of the following values:

`"Many Answers" (Secondaries run BIND 8/9)`: includes as many records as the packet size allows.

`"Many Answers" Except for these servers`: includes as many records as the packet size allows.

`"One Answer" (Secondaries run BIND 4)`: includes one record per packet.

`"One Answer" Except for these servers`: includes one record per packet.

If you specified servers in the transfer_excluded_servers method, the Many Answers or One Answer zone format (depending on transfer_format method setting) will allow zone transfers, except from the specified servers.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting transfer_format
$memberdns->transfer_format( );
#Modifying transfer_format
$memberdns->transfer_format("many_answers");
```

use_lan2_port( )

Use this method to enable/disable DNS service on the LAN2 port.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable DNS service on the LAN2 port or "false" to disable it. Default is "false"

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting use_lan2_port
$memberdns->use_lan2_port( );
#Modifying use_lan2_port
$memberdns->use_lan2_port("true");
```

use_mgmt_port( )

Use this method to enable/disable DNS services on the MGMT port.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable DNS services on the MGMT port or "false" to disable it. Default is "false"

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting use_mgmt_port
$memberdns->use_mgmt_port( );
#Modifying enable_dns
$memberdns->use_mgmt_port("true");
```

use_root_name_servers( )

Use this method to enable/disable custom or Internet root name servers. This setting overrides the grid root name server settings. Specify "False" to indicate that the member overrides the grid root server setting and uses Internet root name servers. Specify "True" to indicate that the member overrides the grid root server setting and uses custom root name servers. Specify "undef" to indicate that the member does not override the grid root server settings.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to allow custom root name servers. Specify "false" to use Internet root name servers. Set it to undef to use the grid root name server settings.

Returns

If you specify a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you do not specify a parameter, the method returns the attribute value.

Example

```
#Get use_root_name_servers
my $useRootNS = $memberdns->use_root_name_servers( );
#Modify use_root_name_servers
$memberdns->use_root_name_servers("false");
#Use grid root server settings
$memberdns->use_root_name_servers(undef);
```

views( )

Use this method to specify or retrieve the view of this member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value. The default value is the "default" view, which means the DNS member object is located under the default view.

Parameter

Valid value is an array reference that contains the names of the DNS views associated with the member. The names must be in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get views
my $ref_views = $object->views();
#Modify DNS views, list of DNS view names in string format
$object->views(["view1","view2","view3"]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object,such as modify and get. This sample also includes error handling for the operations.

#Preparation prior to member DNS object operations

```
#PROGRAM STARTS: Include all the modules that will be used

use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

# Get and modify the member DNS property object. Considering the id grid member named
infoblox.localdomain was previously created.
my $object = $session->get(
    object => "Infoblox::Grid::Member::DNS",
    name   => "infoblox.localdomain"
);

 unless ($object) {
      die("get member DNS failed: ",
      $session->status_code() . ":" . $session->status_detail());
  }
 print "member DNS get  successfull\n";

#Modifying the value of the specified object
$object->enable_dns("true");
$object->forward_only("false");
$object->minimal_resp("true");
$object->use_mgmt_port("true");
$object->notify_source_port(53);
$object->query_source_port(53);

 #Apply the changes
 $session->modify($object)
  or die("modify member DNS failed: ",
      $session->status_code() . ":" . $session->status_detail());

print "DNS member object modified successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::View, Infoblox::Session, Infoblox::Grid::Member::DNS, Infoblox::DNS::DnssecTrustedKey, Infoblox::DNS::Ruleset, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::FileDistribution - Grid Member FileDistribution object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - name( )
  - enable_tftp( )
  - tftp_port( )
  - tftp_allow_access( )
  - enable_http( )
  - enable_http_acl( )
  - http_allow_access( )
  - enable_ftp( )
  - ftp_port( )
  - ftp_allow_access( )
  - ftp_login_banner( )
  - ftp_enable_passive_mode( )
  - ftp_enable_file_listing( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::FileDistribution - Grid Member FileDistribution object.

DESCRIPTION

You can configure the file distribution properties of a grid member. You can enable and disable TFTP, HTTP, and FTP services, and set their operational parameters.

CONSTRUCTOR

```
The Infoblox::Grid::Member::FileDistribution object is automatically generated upon the successful
installation of the Infoblox appliance. It does not require manual construction.
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a grid member FileDistribution object.

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

Apply the following attributes to get a specific grid member FileDistribution object:

name - Required. Host name in string format

Example

```
my $object   = $session->get(
   object    => "Infoblox::Grid::Member::FileDistribution",
   name      => "infoblox.localdomain",
    );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use method to modify the enable_tftp Property.
$object->enable_tftp("true");
#Submit modification
my $response = $session->modify( $object );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of the file distribution properties of a grid member.

name( )

Use this method to retrieve the grid name. This is a required argument.

Parameter

None.

Returns

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $file_distribution->name();
```

enable_tftp( )

Use this method to enable or disable the TFTP service of a member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the TFTP service or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_tftp
$file_distribution->enable_tftp( );
#Modifying enable_tftp
$file_distribution->enable_tftp("true");
```

tftp_port( )

Use this method to specify the port to listen on for the TFTP file distribution service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Port number for TFTP file distribution service. Valid values are between 1 and 63999. The default is 69.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting tftp_port
$file_distribution->tftp_port( );
#Modifying tftp_port
$file_distribution->tftp_port("1069");
```

tftp_allow_access( )

Use this method to allow TFTP file distribution access from specified IPv4 addresses and networks.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 addresses and networks. To reject TFTP file distribution access, add '!' as a prefix to IP addresses and networks. Specify "any" to allow any address or set to an empty array to clear the list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting tftp_allow_access
my $file_distribution = $file_distribution->tftp_allow_access( );
#Modifying tftp_allow_access
$file_distribution->tftp_allow_access(["any"]);
#Modifying tftp_allow_access
$file_distribution->tftp_allow_access(["10.0.0.10", "20.0.0.0/24", "!"."30.0.0.30",
"!"."40.10.0.0/24"]);
#Modifying tftp_allow_access
$file_distribution->tftp_allow_access([]);
```

enable_http( )

Use this method to enable or disable the HTTP file distribution service of a member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable HTTP file distribution service or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_http
$file_distribution->enable_http( );
#Modifying enable_http
$file_distribution->enable_http("true");
```

enable_http_acl( )

Use this method to enable or disable the ACLs for the HTTP file distribution service of a member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable ACLs for the HTTP file distribution service or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_http_acl
$file_distribution->enable_http_acl( );
#Modifying enable_http_acl
$file_distribution->enable_http_acl("true");
```

http_allow_access( )

Use this method to allow HTTP file distribution access from specified IPv4 addresses and networks. enable_http and enable_http_acl must be set to 'true' in order for this http_allow_access to be usable. Set to an empty array to clear the list.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 addresses and networks.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting http_allow_access
my $file_distribution = $file_distribution->http_allow_access( );
#Modifying http_allow_access
$file_distribution->http_allow_access(["10.0.0.10", "20.0.0.0/24", "30.0.0.30", "40.10.0.0/24"]);
#Modifying http_allow_access
$file_distribution->http_allow_access([]);
```

enable_ftp( )

Use this method to enable or disable the FTP service of a member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the FTP service or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting enable_ftp
$file_distribution->enable_ftp( );
#Modifying enable_ftp
$file_distribution->enable_ftp("true");
```

ftp_port( )

Use this method to specify the port to listen on for the FTP file distribution service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Port number for the FTP file distribution service. Valid values are between 1 and 63999. The default is 21.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting ftp_port
$file_distribution->ftp_port( );
#Modifying ftp_port
$file_distribution->ftp_port("1022");
```

ftp_allow_access( )

Use this method to allow FTP file distribution access from specified IPv4 addresses and networks.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is an array reference that contains IPv4 addresses and networks. To reject FTP file distribution access, add '!' as a prefix to IP addresses and networks. Specify "any" to allow any address or set to an empty array to clear the list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting ftp_allow_access
my $file_distribution = $file_distribution->ftp_allow_access( );
#Modifying ftp_allow_access
$file_distribution->ftp_allow_access(["any"]);
#Modifying ftp_allow_access
$file_distribution->ftp_allow_access(["10.0.0.10", "20.0.0.0/24", "!"."30.0.0.30",
"!"."40.10.0.0/24"]);
#Modifying ftp_allow_access
$file_distribution->ftp_allow_access([]);
```

ftp_login_banner( )

Use this method to specify the login banner for the FTP file distribution service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The login banner for the FTP file distribution service. Valid values are strings of up to 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting ftp_login_banner
$file_distribution->ftp_login_banner( );
#Modifying ftp_login_banner
$file_distribution->ftp_login_banner("This is my new FTP login banner");
```

ftp_enable_passive_mode( )

Use this method to enable or disable the FTP service in passive mode.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable the FTP service in passive mode or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting ftp_enable_passive_mode
$file_distribution->ftp_enable_passive_mode( );
#Modifying ftp_enable_passive_mode
$file_distribution->ftp_enable_passive_mode("true");
```

ftp_enable_file_listing( )

Use this method to enable or disable file listing for the FTP service.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable file listing or "false" to disable it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting ftp_enable_file_listing
$file_distribution->ftp_enable_file_listing( );
#Modifying ftp_enable_file_listing
$file_distribution->ftp_enable_file_listing("true");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object,such as modify and get. This sample also includes error handling for the operations.

#Preparation prior to member FileDistribution object operations

```
#PROGRAM STARTS: Include all the modules that will be used

use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Get and modify a Member FileDistribution object

```
# Get and modify the member FileDistribution property object. Note that the grid member named
infoblox.localdomain was previously created.
my $object = $session->get(
    object => "Infoblox::Grid::Member::FileDistribution",
    name   => "infoblox.localdomain"
);

 unless ($object) {
      die("get member FileDistribution failed: ",
      $session->status_code() . ":" . $session->status_detail());
  }
 print "member FileDistribution get successful\n";

#Modifying the value of the specified object
$object->enable_tftp("true");
$object->tftp_allow_access(["any"]);
$object->enable_http("true");
$object->enable_http_acl("true");
$object->http_allow_access(["10.0.0.10", "20.0.0.0/24"]);
$object->enable_ftp("true");
$object->ftp_allow_access(["10.0.0.10", "20.0.0.0/24", "!"."30.0.0.30", "!"."40.10.0.0/24"]);
$object->tftp_port(1069);
$object->ftp_port(1022);
$object->ftp_login_banner("My new banner");
$object->ftp_enable_passive_mode("true");

 #Apply the changes
 $session->modify($object)
  or die("modify member FileDistribution failed: ",
      $session->status_code() . ":" . $session->status_detail());

print "FileDistribution member object modified successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::FileDistributionDir, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::Interface - Interface member object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::Member->additional_ip_list( )
  - Infoblox::Grid::Member::DNS->additional_ip_list( )
- METHODS
  - anycast( )
  - cidr( )
  - comment( )
  - enable_bgp( )
  - enable_ospf( )
  - enable_ipv6_auto_config( )
  - gateway( )
  - interface( )
  - ipv4addr( )
  - ipv6addr( )
  - subnet_mask( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::Interface - Interface member object.

DESCRIPTION

An Interface member object represents additional interface information that can be used at the member level. A single IPv6 address may be added to each member by using an Interface object.

CONSTRUCTOR

```
my  $interface = Infoblox::Grid::Member::Interface->new(
    anycast     => "true" | "false",                          #Optional / Default is "false"
    cidr        => $prefix,                                   #Optional / Default is undefined
    comment     => $string,                                   #Optional / Default is undefined
    enable_bgp  => "true" | "false",                          #Optional / Default is "false"
    enable_ospf => "true" | "false",                          #Optional / Default is "false"
    enable_ipv6_auto_config => "true" | "false",              #Optional / Default is "false"
    gateway     => $ipv4addr | $ipv6addr,                     #Optional / Default is undefined
    interface   => "LAN_HA" | "LOOPBACK" | "MGMT" | "LAN2",   #Optional / Default is "LOOPBACK"
    ipv4addr    => $ipv4addr,                                 #Optional / Default is undefined
    ipv6addr    => $ipv6addr,                                 #Optional / Default is undefined
    subnet_mask => $netmask                                   #Optional / Default is undefined
);
```

MODULE METHODS

The following functions are available to be applied to an Interface object.

Infoblox::Grid::Member->additional_ip_list( )

Use this function to specify Interface for the Grid Member. See Infoblox::Grid::Member->additional_ip_list() for parameters and return values.

Example 1

```
#Construct an Interface object
my $interface1 = Infoblox::Grid::Member::Interface->new(
   ipv4addr    => "10.2.3.4",
   anycast     => "true",
   interface   => "LOOPBACK",
   subnet_mask => "255.255.255.255",
   comment     => "this is a loopback interface"
```

```
);

# Configure Interface on the Grid Member object
my $response = $Grid_Member->additional_ip_list([$interface1]);
```

Example 2

```
#Construct an IPv6 Interface object for member addition
my $interface6 = Infoblox::Grid::Member::Interface->new(
   ipv6addr    => "2001::1234",
   gateway     => "2001::1111",
   anycast     => "false",
   interface   => "LAN_HA",
   cidr        => "64",
   comment     => "this is an IPv6 address"
);

# Configure Interface on the Grid Member object
my $response = $Grid_Member->additional_ip_list([$interface6]);
```

Example 3

```
#Construct an IPv6 Interface object for member addition with IPv6 auto-configuration turned on
my $interface6 = Infoblox::Grid::Member::Interface->new(
   ipv6addr    => "2001::1234",
   enable_ipv6_auto_config  => "true",
   anycast     => "false",
   interface   => "LAN_HA",
   cidr        => "64",
   comment     => "this is an IPv6 address with auto-configuration turned on"
);

# Configure Interface on the Grid Member object
my $response = $Grid_Member->additional_ip_list([$interface6]);
```

Infoblox::Grid::Member::DNS->additional_ip_list( )

Use this function to specify Interface for the Grid Member DNS. See Infoblox::Grid::Member::DNS->additional_ip_list() for parameters and return values.

Example

```
#Construct an Interface object
my $interface2 = Infoblox::Grid::Member::Interface->new(
   ipv4addr    => "11.22.33.44",
   anycast     => "true",
   interface   => "LOOPBACK",
   subnet_mask => "255.0.0.0",
   comment     => "this is a loopback interface"
);

# Configure Interface on the Grid Member DNS object
my $response = $Grid_Member_DNS->additional_ip_list([$interface2]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of an Interface object.

anycast( )

Use this method to enable/disable anycast of the Interface object.

When anycast is enabled the Interface is served for anycast usage.

Parameter

Specify "true" to enable anycast or "false" to disable it. Default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get anycast
my $anycast = $interface->anycast();
#Modify anycast
```

```
$interface->anycast("false");
```

cidr( )

Use this method to set or retrieve the CIDR netmask format of the Inteface object. Required only when ipv6addr is specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the netmask in CIDR format (example: "8" should be specified for netmask 255.0.0.0).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get cidr
my $cidr = $interface->cidr();
#Modify cidr
$interface->cidr("16");
```

comment( )

Use this method to set or retrieve a descriptive comment of Interface object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 512 bytes. Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $interface->comment();
#Modify comment
$interface->comment("Modifying the Interface comment");
```

enable_bgp( )

Use this method to set or retrieve the BGP advertisement setting for this interface.

Parameter

Specify "true" to enable BGP advertisements on this interface or "false" to disable them. Enabling advertisements requires the anycast attribute to be set to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get enable_bgp
my $enable_bgp = $interface->enable_bgp();
#Modify enable_bgp
$interface->enable_bgp("true");
```

enable_ospf( )

Use this method to set or retrieve the OSPF advertisement setting for this interface.

Parameter

Specify "true" to enable OSPF advertisement on this interface or "false" to disable it. Enabling advertisement requires the anycast attribute to be set to "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get enable_ospf
my $enable_ospf = $interface->enable_ospf();
#Modify enable_ospf
$interface->enable_ospf("true");
```

enable_ipv6_auto_config( )

Use this method to enable/disable IPv6 auto-configuration of the Interface object.

When IPv6 auto-configuration is enabled, a gateway is not needed since it will be automatically discovered.

Parameter

Specify "true" to enable IPv6 auto-configuration or "false" to disable it. Default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get IPv6 auto-configuration
my $ipv6_autoconfig_enabled = $interface->enable_ipv6_auto_config();
#Modify IPv6 auto-configuration
$interface->enable_ipv6_auto_config("true");
```

gateway( )

Use this method to set or retrieve the gateway of the Interface object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the gateway of the Interface in IPv4 address format or IPV6 address format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2). An IPv6 address is a 128-bit number in colon hexadecimal notation. It consists of eight 16-bit groups of hexadecimal digits separated by colons (example: 12ab:0000:0000:0123:4567:89ab:0000:cdef).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get gateway
my $gateway = $interface->gateway();
#Modify gateway
$interface->gateway("10.1.1.1");
```

interface( )

Use this method to set or retrieve the interface type of this Interface object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the interface type in string format. Valid values are "LOOPBACK", "LAN_HA", "MGMT" or "LAN2". Default value is "LOOPBACK".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get interface
my $interface = $interface->interface();
#Modify interface
$interface->interface("LAN2");
```

ipv4addr( )

Use this method to set or retrieve the IPV4 address used for the Interface object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $interface->ipv4addr();
#Modify ipv4addr
$interface->ipv4addr("2.2.2.2");
```

ipv6addr( )

Use this method to set or retrieve the IPV6 address used for the Interface object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv6 address is a 128-bit number in colon hexadecimal notation. It consists of eight 16-bit groups of hexadecimal digits separated by colons (example: 12ab:0000:0000:0123:4567:89ab:0000:cdef). IPv6 gateway, and an CIDR are used in conjunctions with ipv6addr ( ).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv6addr
my $ipv6addr = $interface->ipv6addr();
#Modify ipv6addr
$interface->ipv6addr("aaaa:1111:bbbb:2222:cccc:3333:dddd:5555");
```

subnet_mask( )

Use this method to set or retrieve the subnet mask of the Interface object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the subnet mask of the Interface in IPv4 address format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 255.0.0.0).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get subnet_mask
my $ipv4_subnet_mask = $interface->subnet_mask();
#Modify subnet_mask
$interface->subnet_mask("255.255.255.255");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an Interface object such as add, modify, and

remove. Also, this sample includes error handling for the operations.

#Preparation prior to Interface object insertion

```
use strict;
use warnings;
use Infoblox;

my $host_name = "infoblox.localdomain";
my $host_ip   = "192.168.1.2";
#Creating a session to Infoblox appliance.
my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
);

unless ($session) {
    die(qq(constructor for session failed: ),
        join(":", Infoblox::status_code(), Infoblox::status_detail()));
}
print "Session created successfully \n";
```

#Create an Interface object

```
#Creating Interface object.
my $interface1 = Infoblox::Grid::Member::Interface->new(
    ipv4addr     => "10.2.3.4",
    anycast      => "false",
    interface    => "LOOPBACK",
    subnet_mask  => "255.255.255.255",
   ) or die(qq(Constructor for interface1 failed: ),
   join(":", Infoblox::status_code(), Infoblox::status_detail()));
print "Interface object constructed successfully \n";
```

#Add Interface object to Grid Member

```
#Getting the member object from appliance through session.
my @result = $session->get(
    object => "Infoblox::Grid::Member",
    name   => $host_name,
    ) or die(qq(Get Grid Member failed: ),
    join(":", $session->status_code(), $session->status_detail()));
print "Get Grid Member successful\n";

my $result = $result[0];
if ($result) {
    #Modifying the value of the Interface method from the member object.
    $result->additional_ip_list([$interface1]);
    #Applying the changes to appliance through session.
    $session->modify($result)
        or die(qq(Modify Grid Member failed: ),
        join(":", $session->status_code(), $session->status_detail()));
}
print "Interface object added to Grid Member successfully\n";
```

#Modify Interface object

```
#Modifying the comment of the Interface object.
$interface1->comment("Modified Comment");

#Apply changes to the Grid Member object.
$result->additional_ip_list([$interface1]);

#Update Grid Member object through the Infoblox session.
$session->modify($result)
  or die("Modify Grid Member object failed: ",
  $session->status_code() . ":" . $session->status_detail());
print "Grid Member object with modified Interface updated to Infoblox appliance successfully\n";
```

#Remove Interface object

```
#Remove the Interface from the Grid Member object.
$result->additional_ip_list([]);

#Update Grid Member object through the Infoblox session.
$session->modify($result)
  or die("Modify Grid Member object failed: ",
```

```
  $session->status_code() . ":" . $session->status_detail());
print "Removed Interface object from Grid Member successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member, Infoblox::Grid::Member::DNS, Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::License - License object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Grid::Member->node1_licenses( ) / Infoblox::Grid::Member->node2_licenses( )
- METHODS
  - key( )
  - expiry_date( )
  - expiry_time( )
  - type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::License - License object.

DESCRIPTION

Licenses are pre-installed on an Infoblox device according to the software packages ordered at the time of purchase. You can upgrade an Infoblox device with the Keystone license. Contact Infoblox Technical Support to get the Keystone license and install it on the device.

CONSTRUCTOR

```
my $license = Infoblox::Grid::Member::License->new(
    key  => $string,    #Required
);
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a License object.

Infoblox::Session->add( )

Use this method to add a License object to the Infoblox device. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $license = Infoblox::Grid::Member::License->new(
    key  => "EQAAAIMhrdkGjSnnmBlgQSA14b16"
);

# Submit for addition
my $response = $session->add( $license );
```

Infoblox::Grid::Member->node1_licenses( ) / Infoblox::Grid::Member->node2_licenses( )

Use this function to retrieve the licenses of a specific Infoblox grid member.

Example

```
#Retrieve the licenses from a grid member
my $licenses = $grid_member->node1_licenses();
```

METHODS

This section describes all the methods that you can use to set or retrieve the attribute values of a License object.

key( )

Use this method to set or retrieve the license key.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the string with the license key of the device.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get key
my $key = $license->key();

#Modify key
$license->key("EQAAAKsmItQcEWiFb6csgab2VeQ7");
```

expiry_date( )

Use this method to retrieve the license expiry date.

Returns

The method returns the attribute value.

Example

```
#Get the license's expiry date
my $key = $license->expiry_date();
```

expiry_time( )

Use this method to retrieve the license expiry time.

Returns

The method returns the attribute value.

Example

```
#Get the license's expiry time
my $key = $license->expiry_time();
```

type( )

Use this method to retrieve the type of license.

Returns

The method returns the attribute value. Valid values are:

```
"DHCP"                  - Enables the Dynamic Host Configuration Protocol (DHCP) service.

"DNS"                   - Enables the Domain Name System (DNS) service.

"KeystoneDVS"           - Enables grid-based global management and reporting for Domain Name System
(DNS), Dynamic Host Configuration Protocol (DHCP), IP address management (IPAM) appliance solutions.

"NIOSMaintenance"       - For Network Identity Operating System (NIOS) which provides nonstop core
network services such as Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP), IP
address management (IPAM).

"LDAP"                  - Enables the Lightweight Directory Access Protocol services.

"KeystoneSup"           - Enables the Grid Maintenance services.

"IPAM"                  - Enables the IP Address Management services.

"TFTP"                  - Enables the Trivial File Transfer Protocol services.

"NTP"                   - Enables the Network Time Protocol services.

"IDealIP"               - Enables the IDeal IP (DNS, DHCP, IPAM) services.

"AnyCast"               - Enables the AnyCast services.

"WEBUI"                 - Enables the Web User Interface
```

Example

```
#Get type
my $type = $license->type();
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to a License object, such as create, modify, and remove an object. This sample code also includes error handling for the operations.

#Preparation prior to a license insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox device
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);

unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Add License object to Infoblox device

```
#Construct a License object

my $license = Infoblox::Grid::Member::License->new(
    key  => "EQAAAIMhrdkGjSnnmBlgQSA14b16"
);

unless ($license) {
   die("Construct License object failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "License object created successfully\n";

#Adding the license object to Infoblox Appliance through session.
$session->add($license)
    or die("Add Lincense failed: ",
           $session->status_code() . ":" . $session->status_detail() . "\nPlease contact Infoblox
Technical Support to get the license key and add them to the device\n");

print "Added License to Infoblox Appliance successfully.\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::OSPF - OSPF member object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::Member->ospf_list( )
- METHODS
  - area_id( )
  - area_type( )
  - authentication_key( )
  - authentication_type( )
  - comment( )
  - cost( )
  - dead_interval( )
  - hello_interval( )
  - interface( )
  - key_id( )
  - retransmit_interval( )
  - transmit_delay( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::OSPF - OSPF (Open Shortest Path First) member object.

DESCRIPTION

A OSPF member represents the configuration for a OSPF area used at the grid member level.

CONSTRUCTOR

```
my  $ospf = Infoblox::Grid::Member::OSPF->new(
    area_id             => $num | $ipv4addr,                        #Required
    interface           => "LAN_HA", "LOOPBACK", "MGMT", "LAN2"      #Required
    area_type           => "standard" | "stub" | "nssa",             #Optional / Default is "standard"
    authentication_key  => $string | undef,                          #Optional / Default is undefined
    authentication_type => "none" | "simple" | "message-digest",      #Optional / Default is "none"
    comment             => $string | undef,                          #Optional / Default is undefined
    cost                => $num | "AUTO",                            #Optional / Default is "AUTO"
    dead_interval       => $num,                                     #Optional / Default is 40
    hello_interval      => $num,                                     #Optional / Default is 10
    key_id              => $num | undef,                             #Optional / Default is 1
    retransmit_interval => $num,                                     #Optional / Default is 5
    transmit_delay      => $num                                      #Optional / Default is 1
);
```

MODULE METHODS

The following functions are available to be applied to a OSPF object.

Infoblox::Grid::Member->ospf_list( )

Use this function to specify OSPF for the Grid Member. See Infoblox::Grid::Member->ospf_list() for parameters and return values.

Example

```
#Construct a OSPF object
my $ospf = Infoblox::Grid::Member::OSPF->new(
    area_id             => "20",
    area_type           => "standard",
    interface           => "LAN_HA",
    authentication_type => "simple",
    authentication_key  => "infoblox"
);
```

```
# Configure OSPF on the Grid Member object
my $response = $Grid_Member->ospf_list([$ospf]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a OSPF object.

area_id( )

Use this method to set or retrieve the area id value of the OSPF object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the area id in a 32-bit integer (range from 0 to 4294967295) format or in IPv4 address format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get area_id
my $area_id = $ospf->area_id();
#Modify area_id
$ospf->area_id(1);
```

area_type( )

Use this method to set or retrieve the OSPF area type.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the area type in string format. Valid values are "standard", "stub" OR "nssa"(Not-so-stubby). Default value is "standard".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get area_type
my $area_type = $ospf->area_type();
#Modify area_type
$ospf->area_type("stub");
```

authentication_key( )

Use this method to set or retrieve the authentication password to use for OSPF. The authentication key is valid only when authentication type is "simple" or "message-digest".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the authentication key in string format. The authentication key can have a maximum of 8 bytes if authentication type is "simple" or it can have a maximum of 16 bytes if the authentication type is "message-digest". Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authentication_key
my $authentication_key = $ospf->authentication_key();
```

```
#Modify authentication_key
$ospf->authentication_key("infoblox");
```

authentication_type( )

Use this method to set or retrieve the authentication type used for the OSPF advertisement.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the authentication type in string format. Valid values are "none", "simple" or "message-digest". Default value is "none".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authentication_type
my $authentication_type = $ospf->authentication_type();
#Modify authentication_type
$ospf->authentication_type("message-digest");
```

comment( )

Use this method to set or retrieve a descriptive comment of the OSPF configuration.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes. Default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $ospf->comment();
#Modify comment
$ospf->comment("Modifying the OSPF comment");
```

cost( )

Use this method to set or retrieve the cost metric associated with the OSPF advertisement. If "AUTO" is provided, cost metric will be generated automatically.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the cost metric in a 16-bit integer (range from 0 to 65535) format or in string format as "AUTO". Default value is "AUTO".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get cost
my $cost = $ospf->cost();
#Modify cost
$ospf->cost(1);
```

dead_interval( )

Use this method to set or retrieve the dead interval value of OSPF. The dead interval describes the time taken in seconds to wait before declaring that the device is unavailable and down.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the dead interval in a 16-bit integer (range from 0 to 65535) format that represents the duration in seconds. Default value is 40.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dead_interval
my $dead_interval = $ospf->dead_interval();
#Modify dead_interval
$ospf->dead_interval(10);
```

hello_interval( )

Use this method to set or retrieve the hello interval value of OSPF. The hello interval specifies how often to send OSPF hello advertisement, in seconds.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the hello interval in a 16-bit integer (range from 0 to 65535) format that represents the duration in seconds. Default value is 10.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get hello_interval
my $hello_interval = $ospf->hello_interval();
#Modify hello_interval
$ospf->hello_interval(20);
```

interface( )

Use this method to set or retrieve the interface that send out OSPF advertisement information.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the interface in string format. Valid value is one of "LAN_HA", "LOOPBACK", "MGMT", "LAN2"

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get interface
my $interface = $ospf->interface();
#Modify interface
$ospf->interface("LAN_HA");
```

key_id( )

Use this method to set or retrieve the hash key identifier to use for "message-digest" authentication. The hash key identifier is valid only when authentication type is "message-digest".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the hash key identifier in a 8-bit integer (range from 0 to 255) format. Default value is 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get key_id
my $key_id = $ospf->key_id();
#Modify key_id
$ospf->key_id(100);
```

retransmit_interval( )

Use this method to set or retrieve the retransmit interval time of OSPF. The retransmit interval describes the time taken in seconds to wait before retransmitting OSPF advertisement.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the retransmit interval in a 16-bit integer (range from 0 to 65535) format that represents the duration in seconds. Default value is 5.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get retransmit_interval
my $retransmit_interval = $ospf->retransmit_interval();
#Modify retransmit_interval
$ospf->retransmit_interval(50);
```

transmit_delay( )

Use this method to set or retrieve the transmit delay value of OSPF. The transmit delay describes the time taken in seconds to wait before sending an advertisement.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the transmit delay in a 16-bit integer (range from 0 to 65535) format that represents the duration in seconds. Default value is 1.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get transmit_delay
my $transmit_delay = $ospf->transmit_delay();
#Modify transmit_delay
$ospf->transmit_delay(60);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an OSPF object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to OSPF object insertion

```
use strict;
use warnings;
use Infoblox;

my $host_name = "infoblox.localdomain";
my $host_ip = "192.168.1.2";
#Creating a session to Infoblox appliance.
my $session = Infoblox::Session->new(
```

```
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
);

unless ($session) {
    die(qq(constructor for session failed: ),
        join(":", Infoblox::status_code(), Infoblox::status_detail()));
}
print "Session created successfully \n";
```

#Create a grid member OSPF object

```
#Creating OSPF object.
my $ospf1 = Infoblox::Grid::Member::OSPF->new(
    area_id             => "20",
    area_type           => "standard",
    interface           => "LAN_HA",
    authentication_type => "simple",
    authentication_key  => "infoblox"
   ) or die(qq(Constructor for ospf1 failed: ),
   join(":", Infoblox::status_code(), Infoblox::status_detail()));
print "OSPF object constructed successfully \n";
```

#Add OSPF object to Grid Member

```
#Getting the member object from appliance through session.
my @result = $session->get(
    object => "Infoblox::Grid::Member",
    name   => $host_name,
    ) or die(qq(Get Grid Member failed: ),
    join(":", $session->status_code(), $session->status_detail()));
print "Get Grid Member successful\n";

my $result = $result[0];
if ($result) {
    #Modifying the value of the ospf method from the member object.
    $result->ospf_list([$ospf1]);
    #Applying the changes to appliance through session.
    $session->modify($result)
        or die(qq(Modify Grid Member failed: ),
        join(":", $session->status_code(), $session->status_detail()));
}
print "OSPF object added to Grid Member successfully\n";
```

#Modify OSPF object

```
#Modifying the area id of the OSPF object.
$ospf1->area_id("50");

#Apply changes to the Grid Member object.
$result->ospf_list([$ospf1]);

#Update Grid Member object through the Infoblox session.
$session->modify($result)
  or die("Modify Grid Member object failed: ",
  $session->status_code() . ":" . $session->status_detail());
print "Grid Member object with modified OSPF updated to Infoblox appliance successfully\n";
```

#Remove OSPF object

```
#Remove the OSPF from the Grid Member object.
$result->ospf_list([]);

#Update Grid Member object through the Infoblox session.
$session->modify($result)
  or die("Modify Grid Member object failed: ",
  $session->status_code() . ":" . $session->status_detail());
print "Removed OSPF object from Grid Member successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::ServiceStatus - Service Status object.

- NAME
- DESCRIPTION
- MODULE METHODS
  - Infoblox::Grid::Member->service_status( ), Infoblox::Grid::Member->node1_service_status( ), Infoblox::Grid::Member->node2_service_status( )
- METHODS
  - description( )
  - status( )
  - service( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::ServiceStatus - Service Status object.

DESCRIPTION

This object is used to report the service status on an Infoblox appliance. All methods are to be considered read-only.

MODULE METHODS

The following functions can be applied to a ServiceStatus object.

Infoblox::Grid::Member->service_status( ), Infoblox::Grid::Member->node1_service_status( ), Infoblox::Grid::Member->node2_service_status( )

Use this function to retrieve a reference to an array of Infoblox::Grid::Member::ServiceStatus objects that represents the service status for an Infoblox appliance.

Example

```
#Retrieve the member object
my $member = $session->get(object => 'Infoblox::Grid::Member', name => 'hostname.com');

# Access its service status
my @services = @{$member->service_status()};
print $services[0]->status();
```

METHODS

This section describes all the methods that you can use to retrieve the attribute value of an appliance's service status.

description( )

Use this method to retrieve the description of the current ServiceStatus object.

Returns

The method returns the attribute value, which depends on the value of the 'service' method:

```
service value (for service_status) returned by this method
==================================================================
DHCP - DHCP service status
DNS - DNS service status
NTP - NTP service status
TFTP -  Percentage of the file distribution storage space that is used
HTTP_FILE_DIST - Percentage of the file distribution storage space that is used
FTP - Percentage of the file distribution storage space that is used
BLOXTOOLS - CPU/memory/disk space usage percentage

service value (for node1/2_service_status) returned by this method
=============================================================================
NODE_STATUS - Node status
CPU1_TEMP - CPU1 temperature in Celsius
CPU2_TEMP - CPU2 temperature in Celsius
DISK_USAGE - Disk usage percentage
ENET_LAN - LAN1 interface IP
ENET_LAN2 - LAN2 interface IP
ENET_HA - HA interface IP
```

```
ENET_MGMT - MGMT interface IP
LCD - State of the LCD process (running/not running)
MEMORY - Memory usage percentage
REPLICATION - HA replication state
DB_OBJECT - Database capacity usage percentage
RAID_DISK1 - Disk status and type
RAID_DISK2 - Disk status and type
RAID_DISK3 - Disk status and type
RAID_DISK4 - Disk status and type
FAN1 - Fan 1 speed
FAN2 - Fan 2 speed
FAN3 - Fan 3 speed
POWER_SUPPLY - Power status
NTP_SYNC - Indicates if there are "reachable" NTP associations
TEMP_SYS - System temperature in Celsius
RAID_BATTERY - RAID battery status
CPU_USAGE - CPU load average
BGP - BGP service status
OSPF - OSPF service status
```

Example

```
#Get description
my $description = $service_status->description();
```

status( )

Use this method to retrieve the service status.

Returns

The method returns the attribute value. Valid values are "Failed", "Warning", "Working", and "Inactive"

Example

```
#Get status
my $status = $service_status->status();
```

service( )

Use this method to retrieve the service identifier.

Returns

The method returns the attribute value. Possible values when the object is retrieved via service_status are:

DHCP, DNS, NTP, TFTP, HTTP_FILE_DIST, AUTH_NAMED, FTP, BLOXTOOLS

Possible values when the object is retrieved via node1/2_service_status are:

NODE_STATUS, DISK_USAGE, ENET_LAN, ENET_LAN2, ENET_HA, ENET_MGMT, LCD, MEMORY, REPLICATION, OSPF, DB_OBJECT, RAID_DISK1, RAID_DISK2, RAID_DISK3, RAID_DISK4, FAN1, FAN2, FAN3, POWER_SUPPLY, NTP_SYNC, TEMP_CPU1, TEMP_CPU2, TEMP_SYS, RAID_BATTERY, CPU_USAGE

Example

```
#Get service
my $service = $service_status->service();
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member/>

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::Member::StaticRoute - Static route member object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::Member->static_routes( )
- METHODS
  - network( )
  - gateway( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::Member::StaticRoute - Static route member object

DESCRIPTION

A StaticRoute object provides static route information for a grid member.

CONSTRUCTOR

```
my $route = Infoblox::Grid::Member::StaticRoute->new(
       "network" => $network,          #Required
       "gateway" => $ip_address,       #Required
);
```

MODULE METHODS

The following functions can be applied to a StaticRoute object.

Infoblox::Grid::Member->static_routes( )

Use this function to specify a StaticRoute object for the grid member. See Infoblox::Grid::Member->static_routes() for parameters and return values.

Example

```
#Get static routes list
my $static_routes = $grid_member->static_routes();
#Modify static routes list
my $route = Infoblox::Grid::Member::StaticRoute->new(
       "network" => "192.168.2.0/24",
       "gateway" => "192.168.1.1",
);
$grid_member->static_routes([$route]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a StaticRoute object.

network( )

Use this method to set or retrieve the network address of a StaticRoute object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the address and netmask of network in CIDR format (example: 192.168.2.0/24).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network address
my $network= $route->network();
#Modify network address
$route-->network("192.168.2.0/24");
```

gateway( )

Use this method to set or retrieve the gateway of the StaticRoute object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the gateway address in IPv4 address format. An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get gateway address
my $gateway= $route->gateway();
#Modify gateway address
$route-->gateway("192.168.2.0");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a StaticRoute object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a StaticRoute object insertion

```
use strict;
use warnings;
use Infoblox;

my $host_name = "infoblox.localdomain";
my $host_ip   = "192.168.1.2";
#Creating a session to the Infoblox appliance.
my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
);

unless ($session) {
    die(qq(constructor for session failed: ),
        join(":", Infoblox::status_code(), Infoblox::status_detail()));
}
print "Session created successfully \n";
```

#Create a StaticRoute object

```
#Creating a StaticRoute object.
my $route = Infoblox::Grid::Member::StaticRoute->new(
       "network" => "192.168.2.0/24",
       "gateway" => "192.168.1.1",
) or die(qq(Constructor for StaticRoute failed: ),
   join(":", Infoblox::status_code(), Infoblox::status_detail()));
print "StaticRoute object constructed successfully \n";
```

#Add a StaticRoute object to the static routes list of the grid member

```
#Getting the Grid Member object from the appliance through a session.
my @result = $session->get(
    object => "Infoblox::Grid::Member",
    name   => $host_name,
    ) or die(qq(Get Grid Member failed: ),
    join(":", $session->status_code(), $session->status_detail()));
```

```
print "Get Grid Member successful\n";

my $result = $result[0];
if ($result) {
    #Modifying the value of the StaticRoute method from the member object.
    $result->static_routes([$route]);
    #Applying the changes to appliance through session.
    $session->modify($result)
        or die(qq(Modify Grid Member failed: ),
        join(":", $session->status_code(), $session->status_detail()));
}
print "StaticRoute object added to Grid Member successfully\n";
```

#Remove the static routes list

```
#Remove the StaticRoute object from the Grid Member object.
$result->static_routes([]);

#Update the Grid Member object through the Infoblox session.
$session->modify($result)
  or die("Modify Grid Member object failed: ",
  $session->status_code() . ":" . $session->status_detail());
print "Removed StaticRoute object from Grid Member successfully\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::Member

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::MSServer - Microsoft Grid Server object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- Infoblox::DNS::Zone Considerations
  - Methods with no effect
  - Applicable methods
- Infoblox::DNS considerations
- Infoblox::Grid::DNS considerations
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - address()
  - comment()
  - disable()
  - login()
  - password()
  - network_view()
  - dns_view()
  - managing_member()
  - read_only()
  - synchronization_interval()
  - logging_mode()
  - version()
  - is_AD()
  - extensible_attributes( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::MSServer - Microsoft (r) Grid Server object.

DESCRIPTION

A grid Microsoft DNS Server object is a single Microsoft DNS server that is managed by a grid member.

CONSTRUCTOR

```
 my $ms_member = Infoblox::Grid::MSServer->new(
     address                  => $ip_or_fqdn,                                  #Required
     comment                  => $string,                                      #Optional / Default value is
empty
     disable                  => "true" | "false"                              #Optional / Default value is
"false"
     extensible_attributes     => { $string => $string | $num,
                                    $string => [ $string | $num, ... ], ... },   #Optional / Default is
undefined
     login                    => $string,                                      #Required
     password                 => $string,                                      #Optional / Default value is
empty
     managing_member           => $Member,                                      #Optional / Default value is
empty
     read_only                => "true" | "false",                             #Optional / Default value is
"true"
     synchronization_interval => $num,                                          #Optional / Default value is
2
     logging_mode             => "minimum" | "normal" | "advanced" | "full",    #Optional / Default value
is "normal"
     network_view             => $NetworkView,                                 #Optional / Default is
"default" network view
     dns_view                 => $DnsView,                                     #Optional / Default is
"default" view
```

```
);
```

Infoblox::DNS::Zone Considerations

The following apply to zones assigned to Microsoft DNS servers:

Methods with no effect

The following methods have no effect on a Microsoft DNS server that is assigned to a zone:

allow_transfer( )
allow_query( )
allow_update( )
allow_update_forwarding( )
bind_check_names_policy( )
create_underscore_zones( )
disable_forwarding( )
forward_only( )
forward_to( )
import_from( )
override_delegated_ttl( )
override_grid_zone_timer( )
override_grid_email( )
override_serial_number( )
override_soa_mname( )
prefix( )
remove_sub_zones( )
soa_negative_ttl( )
shared_record_groups( )

Values set or retrieved using the member that manages the Microsoft DNS server apply only to that grid member if it is assigned to a zone.

Applicable methods

The following methods apply to any grid member or Microsoft DNS server assigned to the zone:

comment( )
delegated_ttl( )
disable( )
email( )
extensible_attributes( )
host_name_restriction_policy( )
locked( )
name( )
ns_group( )
primary( )
secondaries( )
soa_default_ttl( )
soa_expire( )
soa_mname( )
soa_refresh( )
soa_retry( )
soa_serial_number( )
stub_from( )
stub_members( )
zone( )

Infoblox::DNS considerations

The following objects cannot be used with a Microsoft DNS Server object or with a Zone object whose primary server is a Microsoft DNS Server object:

Infoblox::DNS::BulkHost
Infoblox::DNS::DnssecTrustedKey
Infoblox::DNS::Host
Infoblox::DNS::SRG
Infoblox::DNS::RootNameServer
Infoblox::DNS::Sortlist
Infoblox::DNS::TSIGKey
Infoblox::DNS::Record

DNAME (only with Windows Server 2008 and later)

DS, RRSIG, DNSKEY, NSEC, NSEC3, NSEC3PARAM (read-only on Windows Server 2008 R2 and later)

Infoblox::Grid::DNS considerations

The following methods can be applied to DNS zones managed by a Microsoft DNS server:

default_ttl( )
email( )
host_name_restriction_policy( )
zone_deletion_double_confirm( )

The following methods have no effect on any zone managed by a Microsoft DNS server:

allow_bulkhost_ddns( )
allow_gss_tsig_zone_updates( )
allow_transfer( )
allow_query( )
allow_recursive_query( )
allow_update( )
bind_check_names_policy( )
blackhole_list( )
bulk_host_name_template( )
bulk_host_name_templates( )
dnssec_enabled( )
dnssec_expired_signatures_enabled( )
dnssec_ksk_algorithm()
dnssec_ksk_rollover_interval()
dnssec_ksk_size()
dnssec_signature_expiration( )
dnssec_validation_enabled( )
dnssec_trusted_keys( )
dnssec_zsk_algorithm()
dnssec_zsk_rollover_interval()
dnssec_zsk_size()
enable_blackhole( )
expire_after( )
forward_only( )
forward_updates( )
forwarders( )
logging_categories( )
logging_facility( )
member_secondary_notify( )
negative_ttl( )
notify_source_port( )
nsgroup_default( )
nsgroups( )
query_source_port( )
record_name_policies( )
recursive_query_list( )
refresh_timer( )
retry_timer( )
rootNS( )
sortlist( )
transfer_excluded_servers( )
transfer_format( )

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a grid member object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct an object
my $grid_msserver = Infoblox::Grid::MSServer->new(
                                        address                 => '10.0.0.1',
                                        comment                 => 'This is an MS DNS Server',
                                        disable                 => 'true',
                                        extensible_attributes    => { Site => 'Main Office'},
                                        login                   => 'user',
```

```
                                                    password                 => 'password',
                                                    managing_member          => 'managing.member.com',
                                                    read_only                => 'false',
                                                    synchronization_interval => 5,
                                                    logging_mode             => 'minimum',
    );
#Submit for addition
my $response = $session->add( $grid_msserver );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific grid member object:

 address - Optional. A member address or name in FQDN format.
 comment - Optional. A comment in string format
 read_only - Optional. A boolean value
 managing_member - Optional. A name in FQDN format.
 network_view  -  Optional. The network view served by the member.
 dns_view -  Optional. The DNS view served by the member.
 extensible_attributes - Optional. A hash reference containing extensible attributes.
```

Example

```
my @retrieved_objs = $session->get(
    object  => "Infoblox::Grid::MSServer",
    address => "10.0.0.1");
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment of a Microsoft DNS server managed by the grid.
$grid_msserver->comment("This is modified object");
# Submit modification
my $response = $session->modify( $grid_msserver );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Example

```
#Get the objects with the same name
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::MSServer",
    name   => "10.0.0.1");
#Find the desired object from the retrieved list.
my $desired_gridmsserver = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_gridmsserver );
```

Infoblox::Session->search( )

Use this method to search for Microsoft DNS Server objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a Microsoft DNS Server object

 address - Optional. A member address or name in FQDN format (regular expression).
 comment - Optional. A comment in string format (regular expression).
 read_only - Optional. A boolean value
 managing_member - Optional. The FQDN of the grid member that manages the Microsoft DNS server?
 network_view  -  Optional. The network view served by the member.
 dns_view -  Optional. The DNS view served by the member.
 extensible_attributes   - Optional. A hash reference that contains extensible attributes.
```

Example

```
# search for all matching grid msserver objects
```

```
my @retrieved_objs = $session->search(
    object  => "Infoblox::Grid::MSServer",
    extensible_attributes => { Site => 'Local'},
    );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a Microsoft DNS server.

address()

Retrieve or set the IP address or FQDN of the Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The Microsoft DNS server name in FQDN format or IPV4 format

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get address
my $address = $msserver->address();
```

comment()

Retrieve or set a user-defined string related to this Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $msserver->comment();
#Modify comment
$msserver->comment("Modifying the MS DNS server comment");
```

disable()

Retrieve or set the `disable()` flag of this Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable
my $disable = $msserver->disable();
#Modify disable
$msserver->disable("true");
```

login()

Retrieve or set the login name that the grid member uses to connect to the Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

When setting a value that requires a domain name, prepend the domain information followed by a backslash to the username: "domain\username".

Parameter

The valid value is a string that contains the login name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get login
my $login = $msserver->login();
#Modify login
$msserver->login("newuser");
```

password()

Set the password that the grid member uses to connect to the Microsoft DNS server. This field is used only if the login field for this member is defined.

This field can not be retrieved.

Parameter

The valid value is a string that contains the password.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Set the password
$msserver->password("iG&ojG97Y");
```

network_view()

Retrieve or set the name of the network view associated with this member.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a string that contains the name of the network view associated with this member.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network_view
my $network_view = $msserver->network_view();
#Modify network_view
$msserver->network_view("external");
```

dns_view()

Retrieve or set the name of the DNS view that applies to this Microsoft DNS server. If this field is set to undefined, the default view is used. The DNS view must be set to a DNS view linked to the Microsoft DNS server network view.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a string that contains the name of the DNS view associated with the Microsoft DNS server

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get dns_view
my $dns_view = $msserver->dns_view();
#Modify dns_view
$msserver->dns_view("external");
```

managing_member()

Retrieve or set the grid member responsible for the synchronization of this Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a string that contains the name of the grid member.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get managing_member
my $managing_member = $msserver->managing_member();
#Modify managing_member
$msserver->managing_member("example.member.com");
```

read_only()

Retrieve or set the synchronization mode of this Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Setting the read_only mode to true synchronizes all data from the Microsoft DNS server to the grid and forbids any change on the grid that could be synchronized to the Microsoft DNS server.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get read_only
my $read_only = $msserver->read_only();
#Modify read_only
$msserver->read_only("false");
```

synchronization_interval()

Retrieve or set the synchronization interval of this Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The synchronization interval controls how often data on the Microsoft DNS server is synchronized with the grid. The value is specified in minutes.

Parameter

The valid value is a number that specifies the synchronization interval in minutes. The default value is 2 minutes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get synchronization_interval
my $synchronization_interval = $msserver->synchronization_interval();
#Modify synchronization_interval
$msserver->synchronization_interval(5);
```

logging_mode()

Retrieve or set the synchronization logging verbosity of this Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is "minimum", "normal", "advanced", or "full". The default value is "normal".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get logging_mode
my $logging_mode = $msserver->logging_mode();
#Modify logging_mode
$msserver->logging_mode("advanced");
```

version()

Retrieve the version of this Microsoft DNS server. The returned version is a string, such as "Microsoft Windows Server 2003".

Returns

The method returns the attribute value.

Example

```
#Get version
my $version = $msserver->version();
```

is_AD()

Retrieve the Active Directory flag of the Microsoft DNS server. The method returns "true" if the Microsoft DNS server has Active Directory capabilities or "false" if it does not.

Returns

The method returns the attribute value.

Example

```
#Get is_AD
my $is_AD = $msserver->is_AD();
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with a Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
```

```
    my $ref_extensible_attributes = $msserver->extensible_attributes();
    #Modify extensible attributes
    $msserver->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ]
   });
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, get, modify, search and remove. This sample also includes error handling for the operations.

#Preparation prior to getting and modifying a grid msserver object

```
use strict;
use Infoblox;

#refers to Infoblox Appliance IP address
my $host_ip = "192.168.1.2";

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
);
unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

#Create a grid member object

my $grid_member = Infoblox::Grid::Member->new(
    name          => "hostname.com",
    ipv4addr      => "3.0.0.2",
    mask          => "255.0.0.0",
    gateway       => "3.0.0.1",
    comment       => "Script Generated"
    );
unless ($grid_member) {
       die("Construct grid member object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Grid member object created successfully\n";

#Adding the member object to appliance through session
my $response = $session->add( $grid_member );

unless ($response) {
    die("Add grid member failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print"Grid member added successfully \n";

my $msserver = Infoblox::Grid::MSServer->new(
                                            address                  => '10.0.0.1',
                                            comment                  => 'basic member',
                                            disable                  => 'true',
                                            extensible_attributes     => { Site => 'Local'},
                                            login                    => 'user',
                                            password                 => 'password',
                                            managing_member           => 'hostname.com',
                                            read_only                => 'false',
                                            synchronization_interval => 5,
                                            logging_mode             => 'minimum',
                                           );

unless ($msserver) {
       die("Construct grid MS DNS server object failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Grid MS DNS server object created successfully\n";

#Adding the member object to appliance through session
$response = $session->add( $msserver );
```

```
unless ($response) {
    die("Add grid MS DNS server failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print"Grid MS DNS server added successfully \n";
```

#Search for a grid Microsoft DNS server object

```
 my @retrieved_objs = $session->search(
     object => "Infoblox::Grid::MSServer",
     comment => "basic.*");
 my $object = $retrieved_objs[0];

 unless ($object) {
        die("Search grid MS DNS server object failed: ",
                $session->status_code() . ":" . $session->status_detail());
 }
 print "Search grid MS DNS server object found at least 1 matching entry\n";
```

#Get and modify the msserver object

```
#Get grid member through session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::MSServer",
    address => "10.0.0.1"
    );
my $msserver = $retrieved_objs[0];

unless ($msserver) {
    die("Get grid MS DNS server object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print"Get grid MS DNS server successful \n";

# Modify one of the attributes of the selected Microsoft DNS server.

$msserver->comment("Modified MS DNS server object");

#Applying the changes
$session->modify($msserver)
   or die("Modify grid MS DNS server failed",
             $session->status_code() . ":" . $session->status_detail());
print"Grid MS DNS server modified successfully \n";
```

#Remove the created objects object

```
#Get grid member through session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::MSServer",
    address => "10.0.0.1"
    );
my $desired_msserver = $retrieved_objs[0];

unless ($desired_msserver) {
    die("Get grid MS DNS server object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print"Get grid member successful \n";

#Submit the object for removal
$session->remove( $desired_msserver )
    or die("Remove grid MS DNS server failed",
            $session->status_code() . ":" . $session->status_detail());
print"Grid MS DNS server removed successfully \n";

#Get grid member through session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::Member",
    name   => "hostname.com"
    );
my $desired_grid_member = $retrieved_objs[0];

unless ($desired_grid_member) {
    die("Get grid member object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
```

```
print"Get grid member successful \n";

#Submit the object for removal
$session->remove( $desired_grid_member )
    or die("Remove grid member failed",
           $session->status_code() . ":" . $session->status_detail());
print"Grid member removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::MSServer::DHCP - Grid Microsoft Server DHCP object.

- NAME
- DESCRIPTION
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
  - Infoblox::Session->modify( )
- METHODS
  - manage_dhcp()
  - enable_dhcp()
  - status()
  - status_detail()
  - status_last_updated()
  - last_sync_ts()
  - last_sync_status()
  - last_sync_detail()
  - extensible_attributes()
  - address()
  - broadcast_address()
  - enable_ddns()
  - ddns_server_always_updates()
  - ms_options( )
  - comment()
  - read_only()
  - name()
  - static_hosts()
  - dynamic_hosts()
  - dhcp_utilization()
  - dhcp_utilization_status()
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::MSServer::DHCP - Grid Microsoft (r) Server DHCP object.

DESCRIPTION

You can use Infoblox::Grid::MSServer::DHCP to configure some of the DHCP properties of a Microsoft DHCP server, including enabling/disabling the DHCP service. The service configurations of a grid are not inherited by this object.

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a Microsoft Server DHCP object.

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific grid member object:

 address - Required. A Microsoft server address or name in FQDN (Fully Qualified Domain Name)
format.
```

Example

```
my @retrieved_objs = $session->get(
    object  => "Infoblox::Grid::MSServer::DHCP",
    address => "10.0.0.1");
```

Infoblox::Session->search( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

Apply the following attributes to search for a specific grid member object:

address - Required. A Microsoft server address or name in FQDN (Fully Qualified Domain Name) format. (Regular expressions are supported)

Example

```
my @retrieved_objs = $session->search(
    object  => "Infoblox::Grid::MSServer::DHCP",
    address => "domain.*");
```

Infoblox::Session->modify()

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify a field of a grid Microsoft DHCP server.
$grid_msserver_dhcp->enable_dhcp('true');
# Submit modification
my $response = $session->modify( $grid_msserver_dhcp );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a grid Microsoft DHCP server.

manage_dhcp()

Retrieve or set the managed state of the Microsoft DHCP server. When setting the managed state, use true to enable or false to disable synchronization of the DHCP server. Setting the status does not stop or start the actual DHCP server, but controls whether it is synchronized.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get manage_dhcp
my $manage_dhcp = $msserver_dhcp->manage_dhcp();
#Modify manage_dhcp
$msserver_dhcp->manage_dhcp("true");
```

enable_dhcp()

Set the DHCP service status of the Microsoft DHCP server. When setting the service status, use true to enable or false to disable the DHCP service on the Microsoft DHCP server. Setting the service status stops or starts the actual DHCP service. This only applies if `manage_dhcp()` is true.

When setting the service status, the Microsoft DHCP server will be started or stopped at the next synchronization.

Include the specified parameter to set the attribute value. This setting is write-only. To get the actual status of the service, use the `status()` method.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dhcp
my $enable_dhcp = $msserver_dhcp->enable_dhcp();
```

```
#Modify enable_dhcp
$msserver_dhcp->enable_dhcp("true");
```

status()

Retrieve the running status of the DHCP service of the Microsoft DHCP server. The running status is "running", "stopped", or "error". The returned status is the latest known status as of the date returned by the `status_last_updated()` attribute.

Returns

The method returns the attribute value.

Example

```
#Get status
my $status = $msserver_dhcp->status();
```

status_detail()

Retrieve the detailed running status of the DHCP service of the Microsoft DHCP server. The returned status is the latest known status.

Returns

The method returns the attribute value.

Example

```
#Get status
my $status = $msserver_dhcp->status();
```

status_last_updated()

Retrieve the time when the DHCP service status of the Microsoft DHCP server was last updated.

Returns

The method returns the attribute value.

Example

```
#Get status_last_updated
my $status_last_updated = $msserver_dhcp->status_last_updated();
```

last_sync_ts()

Retrieve the timestamp of the end of the last synchronization attempt related to all DHCP data, except leases.

Returns

The method returns the attribute value.

Example

```
#Get last_sync_ts
my $last_sync_ts = $msserver_dhcp->last_sync_ts();
```

last_sync_status()

Retrieve the result of the last synchronization attempt, if last_sync_ts is defined. Valid values are "Ok", "Warning", or "Error".

Returns

The method returns the attribute value.

Example

```
#Get last_sync_status
my $last_sync_status = $msserver_dhcp->last_sync_status();
```

last_sync_detail()

Retrieve the detailed result of the last synchronization attempt, if last_sync_ts is defined.

Returns

The method returns the attribute value.

Example

```
#Get last_sync_detail
my $last_sync_detail = $msserver_dhcp->last_sync_detail();
```

extensible_attributes()

Use this method to retrieve the extensible attributes of the associated Grid Microsoft DHCP Server object.

This method is read-only. Use the Infoblox::Grid::MSServer object to set or modify the extensible attributes of the Grid Microsoft DHCP Server object.

Returns

The method returns the attribute value.

Example

```
#Get extensible_attributes
my $extensible_attributes = $msserver_dhcp->extensible_attributes();
```

address()

Use this method to retrieve the IP address or FQDN of the associated Grid Microsoft DHCP Server object.

This method is read-only. Use the Infoblox::Grid::MSServer object to set or modify the Grid Microsoft DHCP Server object address.

Returns

The method returns the attribute value.

Example

```
#Get address
my $address = $msserver_dhcp->address();
```

broadcast_address()

Use this method to set or retrieve the broadcast address of the associated Grid Microsoft DHCP Server object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Returns

The Microsoft DHCP server broadcast address in IPV4 format.

Example

```
#Get address
my $address = $msserver_dhcp->broadcast_address();
#Modify broadcast_address
$msserver_dhcp->broadcast_address("10.0.0.1");
```

enable_ddns()

Retrieve or set the dynamic DNS flag of the Microsoft DHCP server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_ddns
my $enable_ddns = $msserver_dhcp->enable_ddns();
#Modify enable_ddns
$msserver_dhcp->enable_ddns("true");
```

ddns_server_always_updates()

Retrieve or set the "Always update DDNS" flag of the Microsoft DHCP server. Use "true" to always allow dynamic DNS updates, or use "false" to only update when requested by the client. The default value is "false".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ddns_server_always_updates
my $ddns_server_always_updates = $msserver_dhcp->ddns_server_always_updates();
#Modify ddns_server_always_updates
$msserver_dhcp->ddns_server_always_updates("true");
```

ms_options( )

Use this method to set or retrieve the custom DHCP options that are associated with the specified Microsoft DHCP server.

Note: You can set only options that are included in both the Microsoft server and NIOS predefined option definitions. Options that are not common to both are read-only and cannot be modified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains defined Infoblox::DHCP::MSOption objects. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get options
my $ref_options = $msserver_dhcp->ms_options();
#Modify options
$msserver_dhcp->ms_options([$option12]); #$option12 is an Infoblox::DHCP::MSOption object
```

comment()

Use this method to retrieve the comment associated with the Grid Microsoft DHCP Server object.

This method is read-only. Use the Infoblox::Grid::MSServer object to set or modify the Grid Microsoft DHCP Server object comment.

Returns

The method returns the attribute value.

Example

```
#Get comment
my $comment = $msserver_dhcp->comment();
```

read_only()

Use this method to retrieve the read_only flag of the associated Grid Microsoft DHCP Server object.

This method is read-only. Use the Infoblox::Grid::MSServer object to set or modify the Grid Microsoft DHCP Server object read_only status.

Returns

The method returns the attribute value.

Example

```
#Get read_only
my $read_only = $msserver_dhcp->read_only();
```

name()

Use this method to retrieve the name of the associated Grid Microsoft DHCP Server object.

This method is read-only. Use the Infoblox::Grid::MSServer object to set or modify the Grid Microsoft DHCP Server object name.

Returns

The method returns the attribute value.

Example

```
#Get name
my $name = $msserver_dhcp->name();
```

static_hosts()

Use this method to retrieve the number of static addresses (fixed addresses and reservations) associated with the Microsoft server.

Returns

The method returns the attribute value.

Example

```
#Get static_hosts
my $static_hosts = $msserver_dhcp->static_hosts();
```

dynamic_hosts()

Use this method to retrieve the number of active leases associated with the Microsoft server.

Returns

The method returns the attribute value.

Example

```
#Get dynamic_hosts
my $dynamic_hosts = $msserver_dhcp->dynamic_hosts();
```

dhcp_utilization()

Use this method to retrieve the DHCP utilization status of the Microsoft server. The returned value is between 0 and 1000. 1000 indicates 100% utilization.

Returns

The method returns the attribute value.

Example

```
#Get dhcp_utilization
my $dhcp_utilization = $msserver_dhcp->dhcp_utilization();
```

dhcp_utilization_status()

Use this method to retrieve the DHCP utilization status of the Microsoft server. The returned value is 'Low', 'Normal', 'High', or 'Full', and depend on the value of the low- and high-water mark parameters.

Returns

The method returns the attribute value.

Example

```
#Get dhcp_utilization_status
my $dhcp_utilization_status = $msserver_dhcp->dhcp_utilization_status();
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::MSServer, Infoblox::DHCP::MSOption

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::MSServer::DNS - Grid Microsoft Server DNS object.

- NAME
- DESCRIPTION
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - manage_dns()
  - enable_dns()
  - status()
  - status_detail()
  - status_last_updated()
  - last_sync_ts()
  - last_sync_status()
  - last_sync_detail()
  - extensible_attributes()
  - address()
  - forwarders()
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::MSServer::DNS - Grid Microsoft (r) Server DNS object.

DESCRIPTION

A Grid Microsoft Server DNS object is a single Microsoft DNS server managed by a grid member.

You can use Infoblox::Grid::MSServer::DNS to configure some of the DNS properties of a Microsoft DNS server, including enabling/disabling the DNS service. The service configurations of a grid are not inherited by this object.

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to a Microsoft Server DNS object.

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific grid member object:

 address - Required. A member address or name in FQDN (Fully Qualified Domain Name) format.
```

Example

```
my @retrieved_objs = $session->get(
    object  => "Infoblox::Grid::MSServer::DNS",
    address => "10.0.0.1");
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment of a grid Microsoft DNS server.
$grid_msserver_dns->enable_dns('true');
# Submit modification
my $response = $session->modify( $grid_msserver_dns );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a grid Microsoft DNS server.

manage_dns()

Retrieve or set the managed state of the Microsoft DNS server. When setting the managed state, use true to enable or false to disable synchronization of the DNS server. Setting the status does not stop or start the actual DNS server but controls whether it is synchronized.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get manage_dns
my $manage_dns = $msserver_dns->manage_dns();
#Modify manage_dns
$msserver_dns->manage_dns("true");
```

enable_dns()

Set the DNS service status of the Microsoft DNS server. When setting the service status, use true to enable or false to disable the DNS service on the Microsoft DNS server. Setting the service status stops or starts the actual DNS service. This only applies if `manage_dns()` is true.

When setting the service status, the Microsoft DNS server will be started or stopped at the next synchronization.

Include the specified parameter to set the attribute value. This setting is read-only. To get the actual status of the service, use the `status()` method.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_dns
my $enable_dns = $msserver_dns->enable_dns();
#Modify enable_dns
$msserver_dns->enable_dns("true");
```

status()

Retrieve the running status of the DNS service of the Microsoft DNS server. Returns a status that is "running", "stopped", or "error". The returned status is the latest known status as of the date returned by the `status_last_updated()` attribute.

Returns

The method returns the attribute value.

Example

```
#Get status
my $status = $msserver_dns->status();
```

status_detail()

Retrieve the detailed running status of the DNS service of the Microsoft DNS server. The returned status is the latest known status.

Returns

The method returns the attribute value.

Example

```
#Get status
my $status = $msserver_dns->status();
```

status_last_updated()

Retrieve the time when the DNS service status of the Microsoft DNS server was last updated.

Returns

The method returns the attribute value.

Example

```
#Get status_last_updated
my $status_last_updated = $msserver_dns->status_last_updated();
```

last_sync_ts()

Retrieve the timestamp of the end of the last synchronization attempt related to zone list and DNS properties.

Returns

The method returns the attribute value.

Example

```
#Get last_sync_ts
my $last_sync_ts = $msserver_dns->last_sync_ts();
```

last_sync_status()

Retrieve the result of the last synchronization attempt, if last_sync_ts is defined. Valid values are "Ok", "Warning", or "Error".

Returns

The method returns the attribute value.

Example

```
#Get last_sync_status
my $last_sync_status = $msserver_dns->last_sync_status();
```

last_sync_detail()

Retrieve the detailed result of the last synchronization attempt, if last_sync_ts is defined.

Returns

The method returns the attribute value.

Example

```
#Get last_sync_detail
my $last_sync_detail = $msserver_dns->last_sync_detail();
```

extensible_attributes()

Use this method to retrieve the extensible attributes of the associated Grid Microsoft DNS Server object.

This method is read-only. Use the Infoblox::Grid::MSServer object to set or modify the extensible attributes of the Grid Microsoft DNS Server object.

Returns

The method returns the attribute value.

Example

```
#Get extensible_attributes
my $extensible_attributes = $msserver_dns->extensible_attributes();
```

address()

Use this method to retrieve the IP address or FQDN of the associated Grid Microsoft DNS Server object.

This method is read-only. Use the Infoblox::Grid::MSServer object to set or modify the IP address of the Grid Microsoft DNS Server object.

Returns

The method returns the attribute value.

Example

```
#Get address
my $address = $msserver_dns->address();
```

forwarders()

Use this method to specify forwarders for the Microsoft DNS server. A forwarder is essentially a name server to which other name servers first send all of their off-site queries. The forwarder builds up a cache of information, avoiding the need for the other name servers to send queries off-site.

This setting is a per Microsoft DNS server setting. It doesn't override the grid-level setting and the grid-level setting does not apply if no value is defined for this Microsoft DNS server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify an array of IPV4/IPV6 addresses to set the list of forwarders.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get forwarders
my $forwarders = $msserver_dns->forwarders();
#Modify forwarders
$msserver_dns->forwarders(['10.0.0.1','10.0.0.2']);
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::MSServer

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::NatGroup - NAT Group object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid->nat_group_list( )
- METHODS
  - comment( )
  - name( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::NatGroup - NAT Group object.

DESCRIPTION

NAT Groups are necessary if the Grid master is behind a NAT device and there memberies behind both side of the NAT device. Any member on the same side as the master go into the same NAT group as the master and use their interface address for Grid communication with each other. Grid members on the other side of that NAT device do not go into the same NAT group as the master and use master's NAT address for Grid communication.

CONSTRUCTOR

```
#Constructor for an NAT Group object
my $nat_group = Infoblox::Grid::NatGroup->new (
        name    => $string      #Required
        comment => $string      #Optional / Default is empty
    );
```

MODULE METHODS

The following functions are available to be applied to an NatGroup object

Infoblox::Grid->nat_group_list( )

Use this function to specify a NatGroup. See Infoblox::Grid->nat_group_list() for parameters and return values.

Example

```
# Retrieve the list of NatGroup objects from Grid object.
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid",
    name   => "Infoblox" );

my $Grid_obj = $retrieved_objs[0];

# construct a NatGroup object
my $nat_group = Infoblox::Grid::NatGroup->new (
        name => "my_nat_group",
        comment => "inside NAT device");

# Modify Nat Group list.
$Grid_obj->nat_group_list([$nat_group]);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute value of a NatGroup.

comment( )

Use this method to set or retrieve the descriptive comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $nat_group->comment();
#Modify comment
$nat_group->comment("new comment");
```

name( )

Use this method to set or retrieve a NAT group name

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get NAT group name
my $nat_group_name = $nat_group->name();
#Modify NAT group name
$nat_group->name("new_group_name");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, modify, and remove an object. This sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Add a NAT Group object

```
my @result_array = $session->get( "object" => "Infoblox::Grid", "name" => "Infoblox");
if( scalar( @result_array ) > 0 )
 {
        my $Grid_obj = $result_array[0];
        unless ($Grid_obj) {
                die("Get Grid object failed: ",
        $session->status_code() . ":" . $session->status_detail());
        }
         # construct a NatGroup object
         my $nat_group = Infoblox::Grid::NatGroup->new (
              name => "my_nat_group",
              comment => "inside NAT device");

         # Add Nat Group to Grid object.
         $Grid_obj->nat_group_list([$nat_group]);
```

```
        $session->modify($Grid_obj);
        print "NAT Group added to Grid object sucessfully\n";
         }
```

#Modify existing NAT Group object

```
@result_array = $session->get( "object" => "Infoblox::Grid", "name" => "Infoblox");
if( scalar( @result_array ) > 0 )
 {
        my $Grid_obj = $result_array[0];
        unless ($Grid_obj) {
                die("Get Grid object failed: ",
        $session->status_code() . ":" . $session->status_detail());
        }

        # get Nat Group from Grid object.
        my $nat_groups_list = $Grid_obj->nat_group_list();
        my $nat_group = @{$nat_groups_list}[0];

        #modify the NAT group name
        $nat_group->name("new_nat_group_name");

        # Add Nat Group to Grid object.
        $Grid_obj->nat_group_list([$nat_group]);

        $session->modify($Grid_obj);
        print "NAT Group modified on Grid object sucessfully\n";
         }
```

#Remove existing NAT Group object

```
@result_array = $session->get( "object" => "Infoblox::Grid", "name" => "Infoblox");
if( scalar( @result_array ) > 0 )
 {
        my $Grid_obj = $result_array[0];
        unless ($Grid_obj) {
                die("Get Grid object failed: ",
        $session->status_code() . ":" . $session->status_detail());
        }

        # empty Nat Group list on Grid object.
        $Grid_obj->nat_group_list([]);

        $session->modify($Grid_obj);
        print "NAT Group removed on Grid object sucessfully\n";
         }

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::NTPAccess - Grid Network Time Protocol access object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid->ntp_access_list( )
- METHODS
  - address( )
  - service( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::NTPAccess - Grid Network Time Protocol (NTP) access object.

DESCRIPTION

The Grid NTP access object is used to specify the access list for the NTP service.

CONSTRUCTOR

```
#Constructor for an Grid NTP access object
 my $ntp_access = Infoblox::Grid::NTPAccess->new(
    address        =>  $ipv4addr | $ipv6addr | $fqdn | "Any"  #Required
    service        => "Time" | "Time and NTP control"         #Optional / Default "Time"
   );
```

This constructed object can be used only when NTP is set to "true" at the grid level.

MODULE METHODS

This section describes all the functions that can be applied to an NTP access object.

Infoblox::Grid->ntp_access_list( ) =head2 Infoblox::Grid::Member->ntp_access_list( )

Use this function to specify the NTP access for the grid/member.

Example

```
# Create NTP access objects
my $ntpac1 = Infoblox::Grid::NTPAccess->new(
                                            address => "2.2.2.2",
                                            service => "Time",
                                           );

my $ntpac2 = Infoblox::Grid::NTPAccess->new(
                                            address => "4.4.4.4",
                                            service => "Time and NTP control",
                                           );

$grid->ntp_access_list([$ntpac1,$ntpac2]);

my $grid = $session->get(
   object  => "Infoblox::Grid",
   name    => "Infoblox"
 );

 unless ($grid) {
     die("get Grid failed:" ,
         $session->status_code(), $session->status_detail());
 }

$grid->ntp_access_list([$ntpac1,$ntpac2]);

#Applying the changes to appliance through session.
$session->modify($grid)
```

```
or die("modify Grid failed:" ,
 $session->status_code(), $session->status_detail());
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute value of an NTP access object.

address( )

Use this method to set or retrieve the IPv4 or IPv6 address or the FQDN of the external NTP client. Use "Any" to apply this permission to all clients.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are an IPv4 or IPv6 address, the FQDN of the external NTP client, or "Any".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get address
my $address = $ntp_access->address();
#Modify address
$ntp_access->address("4.4.4.4");
```

service( )

Use this method to specify to which service this NTP Access object applies.

Parameter

Valid values are "Time" or "Time and NTP control".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get service
my $service = $ntp_access->service();
#Modify service
$ntp_access->service("Time and NTP control");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Grid::Member, Infoblox::Grid, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::NTPKey - Grid Network Time Protocol key object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid->ntp_authentication_key( )
  - Infoblox::Grid::Member->ntp_authentication_key( )
- METHODS
  - key_number( )
  - key_string( )
  - key_type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::NTPKey - Grid Network Time Protocol (NTP) key object.

DESCRIPTION

Grid NTPKey object is used by NTP server to authenticate the clients.

CONSTRUCTOR

```
#Constructor for an Grid NTPKey object
 my $ntp_key = Infoblox::Grid::NTPKey->new(
    key_number   =>  $num,                  #Required
    key_string   =>  $string,               #Required
    key_type     =>  "M" | "S" | "A" | "N"  #Optional / Default is "M"
  );
```

MODULE METHODS

The following functions are available to be applied to an NTPKey object.

Infoblox::Grid->ntp_authentication_key( )

Use this function to specify a NTP key at the grid level on the Infoblox appliance. See Infoblox::Grid->ntp_authentication_key() for parameters and return values.

Example

```
#Create a NTPKey object.
my $ntp_key = Infoblox::Grid::NTPKey->new(
                key_number => 1111,
                key_type   => "M",
                key_string => "aaaa"
              );

#Configure NTPKey on the Infoblox grid object
my $responce = $grid->ntp_authentication_key()([$ntp_key]);
```

Infoblox::Grid::Member->ntp_authentication_key( )

Use this function to specify a NTP key at the member level on the Infoblox appliance. See Infoblox::Grid::Member->ntp_authentication_key() for parameters and return values.

Example

```
#Create a NTPKey object.
 my $ntp_key = Infoblox::Grid::NTPKey->new(
                key_number => 1111,
                key_type   => "M",
                key_string => "aaaa"
              );
```

```
#Configure NTPKey on the Infoblox member object
my $responce = $member->ntp_authentication_key()([$ntp_key]);
```

METHODS

This section describes all the methods that you can used to configure and retrieve the attribute value of a NTPKey object.

key_number( )

Use this method to set or retrieve the key number of the NTPKey object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A positive integer number. Value must be between 1 and 65534.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get key_number
my $number = $ntp_key->key_number();
#Modify key_number
$ntp_key->key_number("3333");
```

key_string( )

Use this method to set or retrieve the key string of the NTPKey object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired key_string in string format defined in the key_type.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get key_string
my $string = $ntp_key->key_string();
#Modify key_string
$ntp_key->key_string("d3e54352e5548080");
```

key_type( )

Use this method to set or retrieve the key type of the NTPKey object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are :"M","S","N","A".The default value is "M".

```
 "M" - For  Message Digest (MD5) in ASCII format and the key_string must be a 1-31 character ASCII
using
          MD5;
 "S" - For Data Encryption Standard (DES) in hex format and key_string must be a 64-bit hexadecimal
        number in DES. The high order 7 bits of each octet form the 56-bit key, and the low order
bit of each
       octet is given a value,  so that the octet maintains odd parity. You must specify leading
zeros so the key
       is exactly 16 hexadecimal digits long and maintains odd parity;

 "A" - For Data Encryption Standard (DES) in ASCII format and the key_string must be a 1-8 character
         ASCII using DES;
 "N" - For Data Encryption Standard (DES) in NTP format and the key_string must be a 64-bit
hexadecimal in
         NTP format. It is the same as the "S" format, but the bits in each octet have been rotated
```

```
one bit right,
          so the parity bit is in the high order bit of the octet. You must specify leading zeros and
odd parity must
          be maintained.
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get key_type
my $type = $ntp_key->key_type();
#Modify key_type
$ntp_key->ntp_key("N");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, modify, and remove an object. Also this sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
 use strict;
 use Infoblox;

 #Create a session to the Infoblox appliance
 my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
 );
 unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
 }
 print "Session created successfully\n";
```

#Adding an NTPKey object

```
#Create a NTPKey object.
my $ntp_key = Infoblox::Grid::NTPKey->new(
   key_number => 65533,
   key_type   => "M",
   key_string => "aaaa"
);

unless($ntp_key) {
       die("Construct NTP Key object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "NTP Key object created successfully\n";

#Create a NTPServer object.
 my $ntp_server = Infoblox::Grid::NTPServer->new(
    address        => "3.3.3.3",
);

unless($ntp_server) {
       die("Construct NTP server object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "NTP Server object created successfully\n";

my @grid_objs = $session->get(
   object  => "Infoblox::Grid",
   name    => "Infoblox"
 );

 my $grid = $grid_objs[0];
 unless ($grid) {
       die("Get Grid object failed: ",
            $session->status_code() . ":" . $session->status_detail());
  }
  print "Get Grid object found at least 1 matching entry\n";
```

```
$grid->enable_ntp("true");
$grid->ntp_authentication_key([$ntp_key]);
$grid->ntp_server([$ntp_server]);

#Applying the changes to appliance through session.
$session->modify($grid)
     or die("modify Grid failed:" ,
      $session->status_code(), $session->status_detail());
print "Grid modified successfully for adding NTPKey object\n";
```

#Modifying an NTPKey object

```
my $ntp_key = Infoblox::Grid::NTPKey->new(
   key_number => 65532,
   key_type   => "M",
   key_string => "aaaa"
);

my @grid_objs = $session->get(
           object => "Infoblox::Grid",
           name   => "Infoblox"
       );

my $grid = $grid_objs[0];
unless ($grid) {
       die("Get Grid object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get Grid object found at least 1 matching entry\n";

$grid->ntp_authentication_key([$ntp_key]);

#Applying the changes to appliance through session.
$session->modify($grid)
     or die("modify Grid failed:" ,
      $session->status_code(), $session->status_detail());
print "Grid modified successfully for modifying NTPKey object\n";
```

#Removing an NTPKey object

```
#Get Infoblox grid object
my @grid_objs = $session->get(
           object => "Infoblox::Grid",
           name   => "Infoblox"
       );

my $grid = $grid_objs[0];
unless ($grid) {
       die("Get Grid object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get Grid object found at least 1 matching entry\n";

$grid->ntp_authentication_key([]);
#Applying the changes to appliance through session.
$session->modify($grid)
    or die("modify Grid failed for removing NTPKey object:" ,
     $session->status_code(), $session->status_detail());
print "Grid modified successfully for removing NTPKey object\n";
####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Grid::NTPServer, Infoblox::Grid,Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::NTPServer - Grid Network Time Protocol server object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid->ntp_server( )
- METHODS
  - address( )
  - authentication( )
  - ntp_key( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::NTPServer - Grid Network Time Protocol (NTP) server object.

DESCRIPTION

The Grid NTP server object is used to synchronize the date and time for an Infoblox appliance.

CONSTRUCTOR

```
#Constructor for an Grid NTP server object
 my $ntp_server = Infoblox::Grid::NTPServer->new(
    address        =>  $ipv4addr | $ipv6addr | $fqdn    #Required
    authentication => "true" | "false",                 #Optional / Default "false"
    ntp_key        => [$NTPKey1, $NTPKey2,...]          #Optional / Default is undefined
   );
```

This constructed object can be used only when NTP is set to "true" at the grid level.

MODULE METHODS

This section describes all the functions that can be applied to a NTP server object.

Infoblox::Grid->ntp_server( )

Use this function to specify the NTP server for the grid.

Example

```
#Create a NTPKey object.
my $ntp_key1 = Infoblox::Grid::NTPKey->new(
   key_number => 1111,
   key_type   => "M",
   key_string => "aaaa"
);

#Create a NTPServer object.
 my $ntp_server = Infoblox::Grid::NTPServer->new(
    address        => "3.3.3.3",
    authentication => "true",
    ntp_key        => $ntp_key1
);

my $grid = $session->get(
   object  => "Infoblox::Grid",
   name    => "Infoblox"
 );

 unless ($grid) {
     die("get Grid failed:" ,
         $session->status_code(), $session->status_detail());
 }
```

```
$grid->enable_ntp("true");
$grid->ntp_authentication_key([$ntp_key1]);
$grid->ntp_server([$ntp_server]);

#Applying the changes to appliance through session.
$session->modify($grid)
     or die("modify Grid failed:" ,
      $session->status_code(), $session->status_detail());
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute value of a NTP server object.

address( )

Use this method to set or retrieve the IPv4 or IPv6 address or FQDN of the external NTP server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are IPv4 or IPv6 address or FQDN of the external NTP server.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get address
my $address = $ntp_server->address();
#Modify address
$ntp_server->address("4.4.4.4");
```

authentication( )

Use this method to enable or disable the authentication for an NTP server.

Parameter

Specify "true" to enable authentication or "false" to disable it. Default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get authentication
my $authentication = $ntp_server->authentication();
#Modify authentication
$ntp_server->authentication("false");
```

ntp_key( )

Use this method to set or retrieve the ntp_key used for authentication.

Parameter

Valid value is an array reference that contains Infoblox::Grid::NTPKey objects.

See Infoblox::Grid::NTPKey for parameters and return values.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Create a NTPKey object.
my $ntp_key2 = Infoblox::Grid::NTPKey->new(
```

```
        key_number => 2222,
        key_type   => "n",
        key_string =>"d3e54352e5548080" );

      #Modify NTPServer object
      my $ntp_server = Infoblox::Grid::NTPServer->new(
         address        => "3.3.3.3",
         authentication => "true",
         ntp_key        => $ntp_key2
      );
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, modify, and remove an object. Also this sample code also includes error handling for the operations.

```
 #PROGRAM STARTS: Include all the modules that will be used
  use strict;
  use Infoblox;

  #Create a session to the Infoblox appliance
  my $session = Infoblox::Session->new(
     master   => "192.168.1.2",
     username => "admin",
     password => "infoblox"
  );
  unless ($session) {
    die("Construct session failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
  }
  print "Session created successfully\n";
```

#Adding an NTP server object #Get the grid object. my $result = $session->get(object => "Infoblox::Grid", name => "Infoblox"); unless ($result) { die("get Grid failed:" , $session->status_code(), $session->status_detail()); }

```
  #Create a NTPKey object.
  my $ntp_key1 = Infoblox::Grid::NTPKey->new(
      key_number => 3333,
      key_type   => "S",
      key_string => "a7cb86a4cba80101"
  );

  #Create a NTPServer object.
  my $ntp_server = Infoblox::Grid::NTPServer->new(
      address        => "3.3.3.3",
      authentication => "true",
      ntp_key        => $ntp_key1
  );
   #Modifying the value of the ntp_authentication_key and ntp_server methods, from the grid object.
  $result->enable_ntp("true");
  $result->ntp_authentication_key([$ntp_key1]);
  $result->ntp_server([$ntp_server]);

  #Applying the changes to appliance through session.
  $session->modify($result)
       or die("modify Grid failed:" ,
        $session->status_code(), $session->status_detail());
   print "\n  Modify Grid successful \n";
```

#Modifying an NTP server object #Create a NTPServer object. my $ntp_server = Infoblox::Grid::NTPServer->new(address => "3.3.3.3", authentication => "false",);

```
 $result->ntp_server([$ntp_server]);

 #Applying the changes to appliance through session.
 $session->modify($result)
      or die("modify Grid failed:" ,
       $session->status_code(), $session->status_detail());
 print "\n  Modify grid successful for NTP server object\n";
```

#Removing an NTP server object #Get the grid object. my $result = $session->get(object => "Infoblox::Grid", name => "Infoblox"); unless ($result) { die("get Grid failed:" , $session->status_code(), $session->status_detail()); }

```
 $result->enable_ntp("false");
 $result->ntp_server([]);
 #Applying the changes to appliance through session.
```

```
$session->modify($result)
     or die("modify Grid failed:" ,
      $session->status_code(), $session->status_detail());
print "\n  Modify grid successful for NTP server object\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Grid::NTPKey, Infoblox::Grid,Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::PersonalSmartFolder - Personal Smart Folder object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - enable_no_values( )
  - group_bys( )
  - name( )
  - query_items( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::PersonalSmartFolder - Personal Smart Folder object.

DESCRIPTION

A Personal Smart Folder object is used for data organization.

CONSTRUCTOR

```
 my $group = Infoblox::Grid::PersonalSmartFolder->new(
     name                => $string,                                                    #Required
     comment             => $string,                                                    #Optional /
Default is undefined
     group_bys           => [$GroupbyItem1, $GroupbyItem2],                              #Optional
/ Default is undefined
     query_items         => [$QueryItem1, $QueryItem2],                                  #Optional
/ Default is undefined
     enable_no_values    => "true" | "false",                                            #Optional
/ Default is "false"
 );
```

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a Personal Smart Folder object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
#Construct the QueryItem in-memory object
my $query_item1 = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                => "location",
    operator            => "eq",
    value               => "Santa Clara",
);

#Construct the PersonalSmartFolder object
my $group_by_state = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                        enable_grouping => 'true',
                                                        value => "State",
                                                        value_type => "EXTATTR",
                                                       );

my $smart_folder = Infoblox::Grid::PersonalSmartFolder->new(
    name                => "smartfolder1",
```

```
    comment              => "personal smart folder",
    group_bys            => [$group_by_state],
    query_items          => [$query_item1],
);
# Submit for addition
my $response = $session->add( $smart_folder );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific PersonalSmartFolder object:

 name      - Required. PersonalSmartFolder name in string format.
 comment   - Optional. Comment in string format.
```

Example

```
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::PersonalSmartFolder",
    name       => "smartfolder1" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment.
$smart_folder->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $smart_folder );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the PersonalSmartFolder objects using a name
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::PersonalSmartFolder",
    name       => "smartfolder1" );
# find the desired object from the retrieved list.
my $desired_smart_folder = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_smart_folder );
```

Infoblox::Session->search( )

Use this method to search for Personal Smart Folder objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a specific PersonalSmartFolder object:

 name      - Optional. PersonalSmartFolder name in string format (regular expression).
 comment   - Optional. Comment in string format (regular expression).
```

Example

```
# search for all PersonalSmartFolder objects that start with "smartfolder"
my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::PersonalSmartFolder",
    name   => "smartfolder.*" );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a PersonalSmartFolder object.

comment( )

Use this method to set or retrieve the comment.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Pertinent information about the personal smart folder. Comment in string format with a maximum of 256 bytes. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $smart_folder->comment();
#Modify comment
$smart_folder->comment("This is the modified comment for smartfolder1");
```

enable_no_values( )

Use this method to set or retrieve the enable_no_values flag. This flag specifies whether objects that do not contain values for the specified group_by attributes are included in the group for the personal smart folder. This flag is disabled by default. To include such objects, enable the option by setting enable_no_values to "true".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get enable_no_values
my $enable_no_values = $smart_folder->enable_no_values();
#Modify enable_no_values
$smart_folder->enable_no_values("true");
```

group_bys( )

Use this method to set or retrieve the group_bys value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::Grid::SmartFolder::GroupBy object(s). The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $group_by_state = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                            enable_grouping => 'true',
                                                            value => "State",
                                                            value_type => "EXTATTR",
                                                           );

#Get group_by value
my $group_by = $smart_folder->group_by();
#Modify group_by value
$smart_folder->group_bys([$group_by_state]);
```

name( )

Use this method to set or retrieve the name of a personal smart folder.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Text with the name of the personal smart folder.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $smart_folder->name();
```

query_items( )

Use this method to set or retrieve the query items. If query items are not specified when you `add()` a smart folder, the server automatically assigns a default query item for the smart folder. This default query item is "type=Network/Zone/Range/Member". This ensures that objects are included in searches for these 4 types, by default. When you `modify()` a smart folder, the server keeps the query items provided by the client and does not add the default items.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::Grid::SmartFolder::QueryItem object(s). The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get query items
my $query_items = $smart_folder->query_items();

#Modify query items
my $query_item1 = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                => "location",
    operator            => "eq",
    value               => "Santa Clara",
);
$smart_folder->query_items([$query_items1]);
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to an PersonalSmartFolder object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
```

```
print "Session created successfully\n";

# Add a network with extensible attributes

my %extensible_attributes=(
    'Region' => 'Santa Clara County',
    'Country' => 'USA',
    'State' => 'California',
    'Site' => 'Santa Clara',
    'Building' => 'HQ',
    'VLAN' => '1',
);

my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/24",
    extensible_attributes => \%extensible_attributes,
);
unless ($network) {
    die("Construct Network failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network created successfully\n";

$session->add($network)
    or die("Add Network object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Network object added to server successfully\n";

#Create a PersonalSmartFolder object

my $query_item = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                     => "Country",
    is_extensible_attribute  => "true",
    operator                 => "eq",
    value                    => "USA",
);

unless ($query_item) {
    die("Construct query item object failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Smart folder query item object created successfully\n";

my $smart_folder = Infoblox::Grid::PersonalSmartFolder->new(
    name                     => "my_folder_personal",
    comment                  => "my personal folder",
    query_items              => [$query_item],
);

unless ($smart_folder) {
    die("Construct personal smart folder object failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Smart folder object created successfully\n";

#Add the PersonalSmartFolder object to the Infoblox appliance through a session
$session->add($smart_folder)
    or die("Add PersonalSmartFolder object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "PersonalSmartFolder object added to server successfully\n";

#Search for a PersonalSmartFolder

my @retrieved_objs = $session->search(
    object => "Infoblox::Grid::PersonalSmartFolder",
    name   => "my_folder.*"
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Search for a PersonalSmartFolder object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search for a PersonalSmartFolder object found at least 1 matching entry\n";

#Get and modify a PersonalSmartFolder object
```

```
#Get the PersonalSmartFolder object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::PersonalSmartFolder",
    name   => "my_folder_personal"
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get PersonalSmartFolder object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get PersonalSmartFolder object found at least 1 matching entry\n";

#Modify the PersonalSmartFolder object
$object->name("my_folder_personal_renamed");
$object->comment("modified personal smart folder");

#Apply the changes.
$session->modify($object)
    or die("Modify PersonalSmartFolder object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "PersonalSmartFolder object modified successfully \n";
```

#Remove a PersonalSmartFolder object

```
#Get the PersonalSmartFolder object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::PersonalSmartFolder",
    name   => "my_folder_personal_renamed"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get PersonalSmartFolder object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get PersonalSmartFolder object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove PersonalSmartFolder object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "PersonalSmartFolder object removed successfully \n";
```

#Cleanup

```
#Get the Network object through the session
my @retrieved_objs = $session->get(
    object  => "Infoblox::DHCP::Network",
    network => "10.0.0.0/24"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove Network object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::SmartFolderChildren, Infoblox::Grid::PersonalSmartFolder, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::Session

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::RecordNamePolicy - Manages the DNS record name policies defined at the grid level.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DNS->record_name_policies( )
- METHODS
  - name( )
  - regex( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::RecordNamePolicy - Manages the DNS record name policies defined at the grid level.

DESCRIPTION

Record name policy defines the DNS record name format restrictions at the grid level.

CONSTRUCTOR

```
my $policy = Infoblox::Grid::RecordNamePolicy->new(
             name  => $string, #Required
             regex => $regex   #Required
);
```

MODULE METHODS

The following functions are available to apply to the record name policy object.

Infoblox::Grid::DNS->record_name_policies( )

Use this function to specify a record name policy at the grid level in the Infoblox appliance.

Example

```
#Create a RecordNamePolicy object.
my $policy = Infoblox::Grid::RecordNamePolicy->new(
                name  => "test",
                regex => "^[a-zA-Z0-9]+\$",
              );

#Configure record name policy on the Infoblox grid object
my $response = $grid_dns->record_name_policies([$policy]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attributes values of a record name policy object.

name( )

Use this method to set or retrieve the record name policy name.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

String with the name of the record name policy.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $policy->name();
#Modify name
$policy->name("new_name");
```

regex( )

Use this method to set or retrieve the record name policy in regular expression format.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The record name policy in regular expression format. It should follow POSIX regular expression rules.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get regex
my $regex = $policy->regex();
#Modify regex
$policy->regex("^[a-zA-Z0-9]*\$");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to a record name policy object, such as create, modify, and remove. This sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a record name policy object

```
my $policy1 = Infoblox::Grid::RecordNamePolicy->new(
    name  => "Allow Any",
    regex => ".+",
    );
unless ($policy1) {
     die("Construct record name policy1 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Record name policy1 object created successfully\n";

my $policy2 = Infoblox::Grid::RecordNamePolicy->new(
      name  => "Allow Underscore",
      regex => "^[-a-zA-Z0-9_]+\$",
    );

unless ($policy2) {
     die("Construct record name policy1 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Record name policy2 object created successfully\n";
```

```
my $policy3 = Infoblox::Grid::RecordNamePolicy->new(
      name  => "Strict Hostname Checking",
      regex => "^[a-zA-Z0-9]\$|^[a-zA-Z0-9][-a-zA-Z0-9]*[a-zA-Z0-9]\$",
    );

unless ($policy3) {
     die("Construct record name policy3 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Record name policy3 object created successfully\n";

my $policy4 = Infoblox::Grid::RecordNamePolicy->new(
    name  => "test",
    regex => "^[a-zA-Z0-9]+\$",
    );
unless ($policy4) {
     die("Construct record name policy3 object failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Record name policy4 object created successfully\n";
```

#Get the Member DNS object to add a record name policy object to it

```
my $grid_dns = $session->get( "object" => "Infoblox::Grid::DNS", "grid" => "Infoblox");

unless ($grid_dns) {
         die("Get Grid DNS failed: ",
             $session->status_code() . ":" . $session->status_detail());
 }
$grid_dns->record_name_policies([$policy1,$policy2,$policy3,$policy4]);
$session->modify($grid_dns)
      or die("modify Grid DNS failed:" ,
      $session->status_code(), $session->status_detail());
print "Modify grid DNS successful for record name policy object\n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::DNS, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::RootNameServer - Custom root name server object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::DNS->rootNS( )
- METHODS
  - host_name( )
  - ipv4addr( )
  - ipv6addr( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::RootNameServer - Custom root name server object.

DESCRIPTION

The custom root name server provides host name and IP address information of internal root name server. When the Infoblox appliance receives a recursive query for DNS data that it does not have, it will use the internal root name server specified, instead of using the internet root name server.

CONSTRUCTOR

```
my $rootns = Infoblox::Grid::RootNameServer->new(
             host_name => $fqdn,      #Required
             ipv4addr  => $ipv4addr   #Required if $ipv6addr is not present
             ipv6addr  => $ipv4addr   #Required if $ipv4addr is not present
);
```

MODULE METHODS

The following functions are available to be applied to a custom root name server object.

Infoblox::Grid::DNS->rootNS( )

Use this function to specify custom root name servers at the grid level on Infoblox appliance. See Infoblox::Grid::DNS->rootNS() for parameters and return values.

Example

```
#Construct a custom root name server object
my $rootns1 = Infoblox::Grid::RootNameServer->new(
            host_name => "qe.infoblox.com",
            ipv4addr  => "1.1.1.1"
);

#Construct a custom root name server object
my $rootns2 = Infoblox::Grid::RootNameServer->new(
            host_name => "doc.infoblox.com",
            ipv6addr  => "2006::132"
);

# Configure custom root name server on the grid DNS object
my $response = $Grid_DNS->rootNS([$rootns1, $rootns2]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute values of a custom root name server.

host_name( )

Use this method to set or retrieve the host name of the custom root name server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify the host name in FQDN (Fully Qualified Domain Name) format. The FQDN consists of the host name followed by the domain name (example: abc.com). A host name can have a maximum of 256 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get host name
my $host_name = $rootns->host_name();
#Modify host name
$rootns->host_name("qa.infoblox.com");
```

ipv4addr( )

Use this method to set or retrieve the IPv4 address of the custom root name server. Required only when ipv6 is not specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192. 168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ipv4addr
my $ipv4addr = $rootns->ipv4addr();
#Modify ipv4addr
$rootns->ipv4addr("2.2.2.2");
```

ipv6addr( )

Use this method to set or retrieve the IPv6 address of the custom root name server. Required only when ipv4 is not specified.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv6 address is a 128-bit number in colon hexadecimal notation. It consists of eight groups of four hexadecimal digits separated by colons (example: 12ab:0000:0000:0123:4567:89ab:0000:cdef).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting ipv6addr
my $ipv6addr = $rootns->ipv6addr();
# Modifying ipv6addr
$rootns->ipv6addr("12ab::345");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a custom root name server object such as add, modify, and remove. Also, this sample includes error handling for the operations.

#Preparation prior to a custom root name server object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;
```

```
#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a custom root name server object

```
my $rootns = Infoblox::Grid::RootNameServer->new(
             host_name => "qe.infoblox.com",
             ipv4addr  => "1.1.1.1"
);
unless($rootns) {
       die("Construct custom root name server object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Root name server object created successfully\n";
```

#Get the Grid DNS object and add custom root name server to it

```
#Get the Grid DNS object
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::DNS",
    grid   => "Infoblox"
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get Grid DNS object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get Grid DNS object found at least 1 matching entry\n";

#Apply the changes to the Grid DNS object
$object->rootNS([$rootns]);

#Apply the changes
$session->modify($object)
    or die("Modify Grid DNS object failed: ",
            $session->status_code() . ":" . $session->status_detail());
print "Grid DNS object with custom root name server updated to Infoblox appliance successfully\n";
```

#Modify custom root name server

```
#Modify existing custom root name server object

#Modifying the host name of the custom root name server object.
$rootns->host_name("eng.infoblox.com");

#Modifying the IPV4 address of the custom root name server object.
$rootns->ipv4addr("3.3.3.3");

#Apply changes to the Grid DNS object.
$object->rootNS([$rootns]);

#Update Grid DNS object through the Infoblox session.
$session->modify($object)
            or die("modify Grid DNS object failed: ",
                $session->status_code() . ":" . $session->status_detail());
       print "Grid DNS object with modified custom root name server updated to Infoblox appliance
successfully\n";
```

#Remove custom root name server

```
#Apply changes to the Grid DNS object to use default internet root name server.
$object->rootNS(undef);

#Update Grid DNS object through the Infoblox session.
$session->modify($object)
            or die("Remove custom root name server from Grid DNS object failed: ",
                $session->status_code() . ":" . $session->status_detail());
       print "Removed custom root name server from Grid DNS object successfully\n";
```

```
####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Grid::DNS

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::ScheduledBackup - Scheduled Backup object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid->scheduled_backup( )
- METHODS
  - backup_type( )
  - backup_frequency( )
  - weekday( )
  - hour_of_day( )
  - minute_of_hour( )
  - user( )
  - password( )
  - backup_server( )
  - path( )
  - disabled( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::ScheduledBackup - Scheduled Backup object.

DESCRIPTION

The Scheduled Backup object is used to schedule system backups.

CONSTRUCTOR

```
#Constructor for a ScheduledBackup object
my $backup = Infoblox::Grid::ScheduledBackup->new (
        backup_type => "LOCAL" | "TFTP" | "FTP" | "SCP",   #Optional / Default is "LOCAL"
        backup_frequency => "Weekly" | "Daily" | "Hourly", #Optional / Default is "Daily"
        weekday => "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "Sunday" | "Monday" | "Tuesday" |
"Wednesday" | "Thursday" | "Friday" | "Saturday"  #Optional / Default is "0"
        hour_of_day      => $num,                          #Optional / Default is 3
        minute_of_hour   => $num,                          #Optional / Default is 0
        user             => $string,                       #Optional / Default is undefined
        password         => $string,                       #Optional / Default is undefined
        backup_server    => $string,                       #Optional / Default is undefined
        path             => $string,                       #Optional / Default is undefined
        disabled         => "true" | "false",              #Optional / Default is "false"
);
```

MODULE METHODS

The following functions can be applied to a ScheduledBackup object.

Infoblox::Grid->scheduled_backup( )

Use this function to specify a ScheduledBackup object. See Infoblox::Grid->scheduled_backup() for parameters and return values.

Example

```
# Retrieve the list of ScheduledBackup objects from the Grid object.
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid",
    name   => "Infoblox"
);

my $Grid_obj = $retrieved_objs[0];

# construct a ScheduledBackup object
my $backup = Infoblox::Grid::ScheduledBackup->new (
    backup_type => "LOCAL",
```

```
        backup_frequency => "Weekly",
        weekday => "0", # "0" is the same as "Sunday"
        hour_of_day => "4",
        minute_of_hour => "0",
        user  => "username",
        password => "user-password",
        backup_server  => "my-backup-server",
        path => "/tmp/infoblox/database.tar.gz",
        disabled => "false",
    );

    # Modify the scheduled backup list
    $Grid_obj->scheduled_backup($backup);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a ScheduledBackup object.

backup_type( )

Use this method to set or retrieve the destination of the backup files. You can back up system files to a local directory, a TFTP server, an FTP server, or an SSH server that supports SCP.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "LOCAL", "TFTP", "FTP" or "SCP". The default value is "LOCAL".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get backup_type
my $backup_type = $backup->backup_type();
#Modify backup_type
$backup->backup_type("FTP");
```

backup_frequency( )

Use this method to set or retrieve the frequency of backups.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "Weekly", "Daily" or "Hourly". The default value is "Daily".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get backup frequency
my $backup_backup_frequency = $backup->backup_frequency();
#Modify backup frequency
$backup->backup_frequency("Weekly");
```

weekday( )

Use this method to set or retrieve the day of the week when the backup is performed. This is required only when backup_frequency is set to "Weekly".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "0", "1", "2", "3", "4", "5", "6", "7", "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday" or "Saturday". "0" means "Sunday", "1" - "Monday". etc. The default value is "6" (Saturday).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get weekday
my $weekday = $backup->weekday();
#Modify weekday
$backup->weekday("0");
```

hour_of_day( )

Use this method to set or retrieve the hour when the backup is performed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A number from 0 through 23. The default value is 3 (3:00 A.M.).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get hour_of_day
my $hour_of_day = $backup->hour_of_day();
#Modify hour_of_day
$backup->hour_of_day("4");
```

minute_of_hour( )

Use this method to set or retrieve the minute of the hour when the backup is performed.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

A number from 0 through 59. The default value is 0.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get minute_of_hour
my $minute_of_hour = $backup->minute_of_hour();
#Modify minute_of_hour
$backup->minute_of_hour("5");
```

user( )

Use this method to set or retrieve the user name on the backup server. This is required only when backup_type is set to "FTP" or "SCP".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The user name in string format with a maximum of 1024 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get user
my $user = $backup->user();
#Modify user
$backup->user("admin1");
```

password( )

Use this method to set or retrieve the user password on the backup server. This is required only when backup_type is set to "FTP" or "SCP".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The password in string format with a maximum of 64 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get password
my $password = $backup->password();
#Modify password
$backup->password("userpassword");
```

backup_server( )

Use this method to set or retrieve the IP address of the backup server. This is required only when backup_type is set to "FTP", "TFTP" or "SCP".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The IP address of the backup server.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get backup_server
my $backup_server = $backup->backup_server();
#Modify backup_server
$backup->backup_server("my-backup-server");
```

path( )

Use this method to set or retrieve the directory path to the backup file stored on the server. This is required only when backup_type is set to "FTP" or "SCP".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired path in string format with a maximum of 1024 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get path
my $path = $backup->path();
#Modify path
$backup->path("/tmp/infoblox/database.tar.gz");
```

disabled( )

Use this method to disable the ScheduledBackup object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to disable the ScheduledBackup object. Set the parameter to "false" to enable the object. The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disabled
my $disabled = $backup->disabled();
#Modify disabled
$backup->disabled("true");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, modify, and remove an object. This sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Add a ScheduledBackup object

```
my @result_array = $session->get( "object" => "Infoblox::Grid", "name" => "Infoblox");
if( scalar( @result_array ) > 0 )
 {
        my $Grid_obj = $result_array[0];
        unless ($Grid_obj) {
                die("Get Grid object failed: ",
        $session->status_code() . ":" . $session->status_detail());
        }
        # construct a ScheduledBackup object
        my $backup = Infoblox::Grid::ScheduledBackup->new (
            backup_type => "LOCAL",
            backup_frequency => "Weekly",
            weekday => "0", # "0" is the same as "Sunday"
            hour_of_day => "4",
            minute_of_hour => "0",
            user  => "username",
            password => "user-password",
            backup_server  => "10.0.3.4",
            path => "/tmp/infoblox/database.tar.gz",
            disabled => "false",
        );

        # Add ScheduledBackup to Grid object.
        $Grid_obj->scheduled_backup($backup);

        $session->modify($Grid_obj)
            or die("Modify Grid failed: ",$session->status_code(). ":" .$session->status_detail());
        print "ScheduledBackup added to Grid object sucessfully\n";
         } else {
             print "No Grid object found.";
         }
```

#Modify an existing ScheduledBackup object

```
@result_array = $session->get( "object" => "Infoblox::Grid", "name" => "Infoblox");
```

```
if( scalar( @result_array ) > 0 )
 {
        my $Grid_obj = $result_array[0];
        unless ($Grid_obj) {
                die("Get Grid object failed: ",
        $session->status_code() . ":" . $session->status_detail());
        }

         # get ScheduledBackup from Grid object.
         my $backup= $Grid_obj->scheduled_backup();

         #modify the backup frequency
         $backup->backup_frequency("Daily");

         # Add ScheduledBackup to Grid object.
         $Grid_obj->scheduled_backup($backup);

         $session->modify($Grid_obj)
            or die("Modify Grid failed: ",$session->status_code(). ":" .$session->status_detail());
         print "ScheduledBackup modified on Grid object sucessfully\n";
          } else {
              print "No Grid object found.";
          }
```

#Remove an existing ScheduledBackup object

```
@result_array = $session->get( "object" => "Infoblox::Grid", "name" => "Infoblox");
if( scalar( @result_array ) > 0 )
 {
        my $Grid_obj = $result_array[0];
        unless ($Grid_obj) {
                die("Get Grid object failed: ",
        $session->status_code() . ":" . $session->status_detail());
        }

         # empty ScheduledBackup list on Grid object.
         $Grid_obj->scheduled_backup([]);

         $session->modify($Grid_obj)
            or die("Modify Grid failed: ",$session->status_code(). ":" .$session->status_detail());
         print "ScheduledBackup removed on Grid object sucessfully\n";
          } else {
              print "No Grid object found.";
          }

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid, Infoblox::Grid->scheduled_backup()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::ScheduledTask - Scheduled Task object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - changed_objects( )
  - scheduled_time( )
  - submit_time( )
  - submitter( )
  - task_id( )
- SAMPLE CODE
- SCHEDULED DATE AND TIME FORMAT
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::ScheduledTask - Scheduled Task object.

DESCRIPTION

A Scheduled Task object is used to retrieve information about scheduled tasks. Scheduled tasks are commands that are scheduled by the Infoblox::Session methods, Infoblox::Session->add(), Infoblox::Session->modify() and Infoblox::Session->remove(), and are called with the scheduled_at parameter.

CONSTRUCTOR

The Infoblox::Grid::ScheduledTask object is automatically generated after the Infoblox::Session methods, Infoblox::Session->add(), Infoblox::Session->modify() and Infoblox::Session->remove(), are called with the scheduled_at parameter. It does not require manual construction.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a Scheduled Task object.

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

Apply the following attributes to get a specific Scheduled Task object:

scheduled_time - The time when the task is scheduled to occur.

scheduled_time_start - Use this with the scheduled_time_end attribute to specify a time range. This is the start time, and scheduled_time_end is the end time. A search returns objects that have scheduled_time attibutes within this time range.

scheduled_time_end - Use this with the scheduled_time_start attribute to specify a time range. This is the end time, and scheduled_time_start is the start time. A search returns objects that have scheduled_time attibutes within this time range.

submit_time - The time when the scheduled task was submitted.

submit_time_start - Use this with the submit_time_end attribute to specify a time range. This is the start time, and submit_time_end is the end time. A search returns objects that have submit_time attibutes within this time range.

submit_time_end - Use this with the submit_time_start attribute to specify a time range. This is the end time, and submit_time_start is the start time. A search returns objects that have submit_time attibutes within this time range.

submitter - The name of the user who submitted the task (regular expression).

task_id - The identifier of a scheduled task.

action - The type of action performed. When used, the search returns ScheduledTask objects that have ScheduledTask::ChangedObject objects in the changed_objects list with matching 'action' attributes. Possible values are:

```
Add
Modify
Delete
Network Discovery
Restart Services
```

object_name - The name of the changed object (regular expression). When used, the search returns ScheduledTask objects that have ScheduledTask::ChangedObject objects in the changed_objects list with matching 'object_name' attirbutes.

object_type - The type of the changed object (regular expression). When used, the search returns ScheduledTask objects that have ScheduledTask::ChangedObject objects in the changed_objects list with matching 'object_type' attributes. Possible values are:

```
A Record
AAAA Record
Authoritative Zone
Bulk Host
CNAME Record
Delegated Zone
DHCP Range
DNAME Record
DNS View
Fixed Address
Forward Zone
Host Record
IPv4 Network
IPv4 Network Container
IPv6 Network
IPv6 Network Container
MX Record
NS Record
PTR Record
Reservation
Roaming Host
Shared A Record
Shared AAAA Record
Shared MX Record
Shared Network
Shared Record Group
Shared SRV Record
Shared TXT Record
SRV Record
Stub Zone
TXT Record
```

Example

```
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::ScheduledTask",
    task_id    => "34" );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the object.
$task->scheduled_time( "2008-10-21T10:56:33Z" );
# Submit modification
my $response = $session->modify( $task );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, use `get()` or `search()` to retrieve the specific object first, and then submit this object for removal.

Example

```
# Get the ScheduledTask objects
my @retrieved_objs = $session->get(
    object     => "Infoblox::Grid::ScheduledTask",
    submitter  => "admin" );
# find the desired object from the retrieved list.
my $task = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $task );
```

Infoblox::Session->search( )

Use this method to search for Admin Group objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

The search can be performed on any combination of the following attributes:

scheduled_time - The time when the task is scheduled to occur.

scheduled_time_start - Use this with the scheduled_time_end attribute to specify a time range. This is the start time, and scheduled_time_end is the end time. A search returns objects that have scheduled_time attibutes within this time range.

scheduled_time_end - Use this with the scheduled_time_start attribute to specify a time range. This is the end time, and scheduled_time_start is the start time. A search returns objects that have scheduled_time attibutes within this time range.

submit_time - The time when the scheduled task was submitted.

submit_time_start - Use this with the submit_time_end attribute to specify a time range. This is the start time, and submit_time_end is the end time. A search returns objects that have submit_time attibutes within this time range.

submit_time_end - Use this with the submit_time_start attribute to specify a time range. This is the end time, and submit_time_start is the start time. A search returns objects that have submit_time attibutes within this time range.

submitter - The name of the user who submitted the task (regular expression).

task_id - The identifier of a scheduled task.

action - The type of action performed. When used, the search returns ScheduledTask objects that have ScheduledTask::ChangedObject objects in the changed_objects list with matching 'action' attributes. Possible values are:

```
Add
Modify
Delete
Network Discovery
Restart Services
```

object_name - The name of the changed object (regular expression). When used, the search returns ScheduledTask objects that have ScheduledTask::ChangedObject objects in the changed_objects list with matching 'object_name' attirbutes.

object_type - The type of the changed object (regular expression). When used, the search returns ScheduledTask objects that have ScheduledTask::ChangedObject objects in the changed_objects list with matching 'object_type' attributes. Possible values are:

```
A Record
AAAA Record
Authoritative Zone
Bulk Host
CNAME Record
Delegated Zone
DHCP Range
DNAME Record
DNS View
Fixed Address
Forward Zone
Host Record
IPv4 Network
IPv4 Network Container
IPv6 Network
IPv6 Network Container
MX Record
NS Record
PTR Record
Reservation
Roaming Host
Shared A Record
Shared AAAA Record
Shared MX Record
Shared Network
Shared Record Group
Shared SRV Record
Shared TXT Record
SRV Record
Stub Zone
TXT Record
```

Example

```
# search for all ScheduledTask objects
my @retrieved_objs = $session->search(
    object  => "Infoblox::Grid::ScheduledTask",
    scheduled_time_start => "2008-10-21T10:56:33Z",
    scheduled_time_end => "2008-11-21T10:56:33Z" );
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of a ScheduledTask object.

changed_objects( )

Use this method to retrieve information about the objects that are scheduled to be changed.

Parameter

none

Returns

The method returns a reference to an array of Infoblox::Grid::ScheduledTask::ChangedObject objects.

Example

```
#Get changed_objects
my $changed_objects = $task->changed_objects();
my $obj = $changed_objects->[0];
```

scheduled_time( )

Use this method to set or retrieve the date and time the task is scheduled to occur.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Scheduled date and time in ISO 8601 extended format (e.g., 2008-10-21T10:56:33Z or 2008-10-21T10:56:33.001Z).

Use case insensitive value "now" to execute a scheduled task immediately.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get scheduled_time value
my $scheduled_time = $task->scheduled_time();

#Modify scheduled_time value
$task->scheduled_time("2008-10-21T10:56:33Z");
# Submit modification
$session->modify($task);

# Do it now
$task->scheduled_time("Now");
# Submit modification
$session->modify($task);
```

submit_time( )

Use this method to retrieve the time the scheduled task was submitted (when appropriate Session method was called with parameter scheduled_time).

Parameter

none

Returns

The method returns the submitted date and time in ISO 8601 extended format (e.g., 2008-10-21T10:56:00Z or 2008-10-21T10:56:00.001Z).

Example

```
#Get submit_time
my $submit_time = $task->submit_time();
```

submitter( )

Use this method to retrieve the name of the user who submitted the task.

Parameter

none

Returns

The method returns the user name in string format.

Example

```
#Get submitter
my $submitter = $task->submitter();
```

task_id( )

Use this method to retrieve the identifier of a scheduled task.

Parameter

none

Returns

The method returns the identifier in numeric format.

Example

```
#Get task_id
my $task_id = $task->task_id();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a ScheduledTask object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

# Add a network
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/24",
);
unless ($network) {
    die("Construct Network failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network created successfully\n";

$session->add($network)
    or die("Add Network object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Network object added to server successfully\n";
```

#Create a ScheduledTask object

```
# Create a fixed address
my $fixed_addr = Infoblox::DHCP::FixedAddr->new(
    ipv4addr    => '10.0.0.3',
    mac         => '11:22:33:44:55:66'
);
unless ($fixed_addr) {
    die("Construct FixedAddr failed: ",
```

```
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "FixedAddr created successfully\n";

# Schedule the addition of a fixed address
$session->add($fixed_addr, scheduled_at => "2020-01-01T14:52:00Z")
    or die("Schedule fixed address add failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Fixed address add scheduled successfully\n";
```

#Search for a ScheduledTask object

```
my @retrieved_objs = $session->search(
    object    => "Infoblox::Grid::ScheduledTask",
    submitter => "admin"
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Search for a ScheduledTask object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search for a ScheduledTask object found at least 1 matching entry\n";
```

#Get and modify a ScheduledTask object

```
#Get the ScheduledTask object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object         => "Infoblox::Grid::ScheduledTask",
    scheduled_time => "2020-01-01T14:52:33Z"
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get ScheduledTask object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get ScheduledTask object found at least 1 matching entry\n";

#Modify the ScheduledTask object
$object->scheduled_time("2020-02-01T14:52:33Z");

#Apply the changes.
$session->modify($object)
    or die("Modify ScheduledTask object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "ScheduledTask object modified successfully \n";
```

#Remove a ScheduledTask object

```
#Get the ScheduledTask object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::ScheduledTask",
    action => "Add"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get ScheduledTask object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get ScheduledTask object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove ScheduledTask object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "ScheduledTask object removed successfully \n";
```

#Cleanup

```
#Get the Network object through the session
my @retrieved_objs = $session->get(
    object  => "Infoblox::DHCP::Network",
    network => "10.0.0.0/24"
);
my $object = $retrieved_objs[0];
unless ($object) {
```

```
    die("Get Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove Network object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

####PROGRAM ENDS####
```

SCHEDULED DATE AND TIME FORMAT

Scheduled date and time are represented in ISO 8601 extended format (e.g., 2008-10-21T10:56:00Z or 2008-10-21T10:56:00.001Z) with some restrictions:

- Maximal value for year field is 2037.

When the scheduling parameter is used, changes to objects do not occur immediately. Instead, the Infoblox::Grid::ScheduledTask object is created, and then executed at the time specified in this parameter.

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::ScheduledTask, Infoblox::Grid::ScheduledTask::ChangedObject, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::ScheduledTask::ChangedObject - Scheduled Task Changed Object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::ScheduledTask->changed_objects( )
- METHODS
  - action( )
  - changed_properties( )
  - object_name( )
  - object_type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::ScheduledTask::ChangedObject - Scheduled Task Changed Object.

DESCRIPTION

A Scheduled Task Changed Object is used to retrieve information about objects with changes that were scheduled in Infoblox::Grid::ScheduledTask.

CONSTRUCTOR

The Infoblox::Grid::ScheduledTask::ChangedObject is a read-only object and does not require manual construction.

MODULE METHODS

The following functions can be applied to a ChangedObject object.

Infoblox::Grid::ScheduledTask->changed_objects( )

Use this function to retrieve the changed objects list. See Infoblox::Grid::ScheduledTask->changed_objects() for parameters and return values.

Example

```
# Retrieve the list of ScheduledTask objects
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::ScheduledTask",
    submitter => "admin" );

my $task = $retrieved_objs[0];

# get ChangedObject
my $changed_objects = $task->changed_objects();
my $obj = $changed_objects->[0];
```

METHODS

This section describes all the methods that you can use to retrieve the attribute values of a ChangedObject object.

action( )

Use this method to retrieve the operation performed on the objects.

Parameter

none

Returns

The method returns the operation in string format. Possible values are:

"Network Discovery" for network_discovery_control( ) service,

"Restart Services" for restart( ) services,

"Add", "Modify", "Delete" for all other objects.

Example

```
#Get action
my $action = $changed_object->action();
```

changed_properties( )

Use this method to retrieve the list of changed properties in the object.

Parameter

none

Returns

The method returns a reference to an array of strings. Each string in the array describes the changes. The descriptions are similiar to the entries in the Change Audit Log column in the GUI.

Example

```
#Get changed_properties value
my $changed_properties = $changed_object->changed_properties();
for (@$changed_properties) {
    print $_,"\n";
}
```

object_name( )

Use this method to retrieve the name of the changed object.

Parameter

none

Returns

The method returns the object name in string format.

Example

```
#Get object_name
my $object_name = $changed_object->object_name();
```

object_type( )

Use this method to retrieve the type of the changed object.

Parameter

none

Returns

The method returns an object type in string format, such as "A Record" or "Shared TXT Record".

Example

```
#Get object_type
my $object_type = $changed_object->object_type();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparation prior to a ScheduledTask object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;
```

```
#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
      die("Construct session failed: ",
              Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

# Add a network
my $network = Infoblox::DHCP::Network->new(
    network => "10.0.0.0/24",
);
unless ($network) {
    die("Construct Network failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network created successfully\n";

$session->add($network)
    or die("Add Network object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Network object added to server successfully\n";
```

#Create a ScheduledTask object

```
# Create a fixed address
my $fixed_addr = Infoblox::DHCP::FixedAddr->new(
    ipv4addr    => '10.0.0.3',
    mac         => '11:22:33:44:55:66'
);
unless ($fixed_addr) {
    die("Construct FixedAddr failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "FixedAddr created successfully\n";

# Schedule the addition of a fixed address
$session->add($fixed_addr, "2020-01-01T14:52:00Z")
    or die("Schedule fixed address add failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Fixed address add scheduled successfully\n";
```

#Search for a ScheduledTask

```
my @retrieved_objs = $session->search(
    object    => "Infoblox::Grid::ScheduledTask",
    submitter => "admin"
);
my $object = $retrieved_objs[0];

unless ($object) {
    die("Search for a ScheduledTask object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Search for a ScheduledTask object found at least 1 matching entry\n";
my $objs = $object->changed_objects;
my $changed_obj = $objs->[0];
my $action = $changed_obj->action();
my $object_name= $changed_obj->object_name();
my $object_type= $changed_obj->object_type();
my $changed_properties= $changed_obj->changed_properties();
my $property1 = $changed_properties->[0];
```

#Get and modify a ScheduledTask object

```
#Get the ScheduledTask object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object         => "Infoblox::Grid::ScheduledTask",
    scheduled_time => "2020-01-01T14:52:00Z"
);
```

```
my $object = $retrieved_objs[0];

unless ($object) {
    die("Get ScheduledTask object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get ScheduledTask object found at least 1 matching entry\n";

#Modify the ScheduledTask object
$object->scheduled_time("2020-02-01T14:52:00Z");

#Apply the changes.
$session->modify($object)
    or die("Modify ScheduledTask object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "ScheduledTask object modified successfully \n";
```

#Remove a ScheduledTask object

```
#Get the ScheduledTask object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::ScheduledTask",
    action => "Add"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get ScheduledTask object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get ScheduledTask object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove ScheduledTask object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "ScheduledTask object removed successfully \n";
```

#Cleanup

```
#Get the Network object through the session
my @retrieved_objs = $session->get(
    object  => "Infoblox::DHCP::Network",
    network => "10.0.0.0/24"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get Network object failed: ",
        $session->status_code() . ":" . $session->status_detail());
}
print "Get Network object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
    or die("Remove Network object failed: ",
    $session->status_code() . ":" . $session->status_detail());
print "Network object removed successfully \n";

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::ScheduledTask, Infoblox::Grid::ScheduledTask->changed_objects()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SmartFolder::GroupBy - Smart Folder Group By object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::GlobalSmartFolder->group_bys( )
  - Infoblox::Grid::PersonalSmartFolder->group_bys( )
- METHODS
  - value( )
  - value_type( )
  - enable_grouping( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SmartFolder::GroupBy - Smart Folder Group By object

DESCRIPTION

Defines grouping information in a global or personal smart folder.

CONSTRUCTOR

```
my $group_by = Infoblox::Grid::SmartFolder::GroupBy->new(
    value                   => $string,            #Required
    enable_grouping         => "true" | "false",   #Required
    value_type              => "EXTATTR" | "NORMAL", #Optional / Default is "EXTATTR"
);
```

MODULE METHODS

The following functions can be applied to a GroupBy object.

Infoblox::Grid::GlobalSmartFolder->group_bys( )

Use this function to specify a list of "group by" items to store in a global smart folder.

Example

```
#Construct an object
my $group_by_state = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                              enable_grouping => 'true',
                                                              value => "State",
                                                              value_type => "EXTATTR",
                                                             );

my $group_by_vcluster = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                            enable_grouping => 'true',
                                                            value => "v_cluster",
                                                            value_type => "NORMAL",
                                                           );
# Configure group_by items
my $response = $GlobalSmartFolder->group_bys([$group_by_state,$group_by_vcluster]);
```

Infoblox::Grid::PersonalSmartFolder->group_bys( )

Use this function to specify a list of "group by" items to store in a personal smart folder.

Example

```
#Construct an object
my $group_by_state = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                              enable_grouping => 'true',
                                                              value => "State",
```

```
                                                                    value_type => "EXTATTR",
                                                                   );

my $group_by_vcluster = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                                  enable_grouping => 'true',
                                                                  value => "v_cluster",
                                                                  value_type => "NORMAL",
                                                                 );
# Configure group_by items
my $response = $PersonalSmartFolder->group_bys([$group_by_state,$group_by_vcluster]);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a SmartFolder::GroupBy object.

value( )

Use this method to specify the name of the attribute (when value_type is "EXTATTR" or "NORMAL"). The possible values for the value attribute are v_datacenter, v_cluster, v_host, v_entity_type, port_vlan_name, network_component_name, network_component_port_name, and any valid extensible attribute in the appliance.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are strings corresponding to extensible attributes, object types, or object names.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting value
my $value = $group_by->value();
# Modifying value
$group_by->value("Country");
```

value_type( )

Use this method to set or retrieve the value type.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

"EXTATTR" and "NORMAL" are supported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting value_type
my $value_type = $group_by->value_type();
# Modifying value_type
$group_by->value_type("EXTATTR");
```

enable_grouping( )

Use this method to set or retrieve the enable grouping attribute.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to set the enable_grouping flag or "false" to deactivate/unset it. "group by" rules where enable_grouping is set to "false" will be ignored when retrieving the corresponding smart folder.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting enable_grouping
my $enable_grouping = $group_by->enable_grouping();
# Modifying enable_grouping
$group_by->enable_grouping("true");
```

SAMPLE CODE

##Preparation prior to a smart folder group by object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a smart folder query item object

```
my $group_by_state = Infoblox::Grid::SmartFolder::GroupBy->new(
                                                            enable_grouping => 'true',
                                                            value => "State",
                                                            value_type => "EXTATTR",
                                                           );
unless ($group_by_state){
     die("Construct smart folder group_by item failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}

print "SmartFolder::GroupBy object created successfully\n";
```

#Create a personal smart folder and use $group_by

```
 #create a personal smart folder and use $query_item
 my $smart_folder = Infoblox::Grid::PersonalSmartFolder->new(
    name         => "my_personal_folder",
    group_bys    => [$group_by_state],
  );

#put smart folder into session
$session->add($smart_folder)
  or die("Smart folder creation failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());

print "Smart folder creation successful\n";
```

#Clean up

```
@result = $session->get(
    object => "Infoblox::Grid::PersonalSmartFolder",
    name   => "my_personal_folder"
);

unless (scalar(@result) == 0) {
    $session->remove($result[0]);
}

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Grid::GlobalSmartFolder, Infoblox::Grid::PersonalSmartFolder

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SmartFolder::GroupByValue - Smart Folder Group By Value object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
- METHODS
  - name( )
  - value( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SmartFolder::GroupByValue - Smart Folder Group By Value object

DESCRIPTION

Defines a group value as part of an Infoblox::Grid::SmartFolderChildren query.

CONSTRUCTOR

```
my $group_by_value = Infoblox::Grid::SmartFolder::GroupByValue->new(
    name                    => $string,            #Required
    value                   => $string,            #Required
);
```

MODULE METHODS

METHODS

This section describes all the methods that you can use to configure a SmartFolder::GroupByValue object.

name( )

Use this method to specify the extensible attribute, object type, or object name that is used to group objects.

Parameter

Valid values are strings corresponding to the names.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

value( )

Use this method to specify the value of the extensible attribute, object type, or object name.

Parameter

Valid values are strings corresponding to the required extensible or normal attribute values.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::SmartFolderChildren

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SmartFolder::QueryItem - Smart Folder Query Item object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::GlobalSmartFolder->query_items( )
- METHODS
  - is_extensible_attribute( )
  - name( )
  - operator( )
  - value( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SmartFolder::QueryItem - Smart Folder Query Item object

DESCRIPTION

Manipulates a Smart Folder Query Item object that contains information about a query to be stored in a global or personal smart folder.

CONSTRUCTOR

```
 my $query_item = Infoblox::Grid::SmartFolder::QueryItem->new(
     name                    => $string,   #Required
     is_extensible_attribute => $string,   #Optional / Default is "true"
     operator                => "eq" | "ne" | "lt" | "gt" | "begins with" | "does not begin with" | "ends
with" | "does not end with" | "has value" | "does not have value" | "within" | "today" | "yesterday" |
"this week" | "this month",   #Required
     value                   => $string | $num | $boolean | type of extensible attribute definition,
#Optional / Default is undefined
 );
```

MODULE METHODS

The following functions can be applied to a query item object.

Infoblox::Grid::GlobalSmartFolder->query_items( )

Use this function to specify a list of query items to store in a global smart folder.

Example

```
#Construct an object
  my $query_item1 = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                  => "Building",
    is_extensible_attibute => "true",
    operator              => "eq",
    value                 => "office1"
);

#Construct an object
  my $query_item2 = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                  => "address",
    is_extensible_attibute => "false",
    operator              => "ne",
    value                 => "10.0.0.1"
);

# Configure query items
my $response = $GlobalSmartFolder->query_items([$query_item1,$query_item2]);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a SmartFolder::QueryItem object.

is_extensible_attribute( )

Use this method to indicate whether this smart folder query item is an extensible or not. When the attribute is_extensible_attribute is "true", the query item is an extensible attribute. When it is "false", it is a normal attribute.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are "true" or "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting is_extensible_attribute
my $is_extensible_attribute = $query_item->is_extensible_attribute();
# Modifying is_extensible_attribute
$query_item->is_extensible_attribute("true");
```

name( )

Use this method to set or retrieve the name of the smart folder query item. The possible values for the name attribute are

```
address
circuit_id
comment
disabled
duid
is_leaf_network
last_discovered_timestamp
locked
mac_address
name
network_component_name
network_component_port_name
network_view
port_vlan_name
record_type
remote_id
server_association_type
service
service_status
type
unmanaged
v_cluster
v_datacenter
v_entity_type
v_host
was_conflict
zone_func
zone_type
```

or any valid extensible attribute definition name present in the system.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired name for the smart folder query item.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting name
my $name = $query_item->name();
# Modifying name
$query_item->name("Region");
```

operator( )

Use this method to set or retrieve the operator used in the query.

Parameter

Valid values are any of the following strings: "eq", "ne", "lt", "gt", "begins with", "does not begin with", "ends with", "does not end with", "has value", "does not have value", "within", "today", "yesterday", "this week", and "this month".

"has_value" and "does not have value" apply to extensible attribute values only.

For valid values for 'value' parameter for the "within" operator, please see the description of the 'value' method.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting operator
my $operator = $query_item->operator();
# Modifying operator
$query_item->operator("today");
```

value( )

Use this method to set or retrieve the value used in the query.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired value in string format.

For the "within" operator, the following format is supported: n [d/w/m], where n is the number of days/weeks/months depending on the unit following it. The unit is required. For example, 4d (4 days), 1w (1 week), or 7m (7 months)

For the following query items, a boolean value 'true' or 'false' will be expected: "disabled", "is_leaf_network", "locked".

For a "type" query item, a valid value is one of:

```
'All'
'AllNetwork'
'AllTemplate'
'DhcpFailoverAssoc'
'DhcpFilter'
'DHCPObjects'
'DhcpRange'
'DhcpRangeTemplate'
'DNSObjects'
'FixedAddress'
'FixedAddressTemplate'
'Host'
'IfmapClient'
'IPAMObjects'
'IPv4DHCPObjects'
'IPv6DhcpFilter'
'IPv6DHCPObjects'
'IPv6DhcpRange'
'IPv6DhcpRangeTemplate'
'IPv6FixedAddress'
'IPv6FixedAddressTemplate'
'IPv6Lease'
'IPv6Network'
'IPv6NetworkTemplate'
'IPv6OptionSpace'
'IPv6SharedNetwork'
'MacFilterAddress'
'Member'
'Network'
'NetworkTemplate'
'Network/Zone/Range/Member'
'NsGroup'
'OptionFilterMatchRule'
'OptionSpace'
'Reservation'
'ResourceRecord'
'RoamingHost'
'SharedNetwork'
'SharedRecordGroup'
'View'
'Zone'
```

For a "record_type" query item a valid value is one of:

```
'A'
'AAAA'
'All types'
'CNAME'
```

```
'DNAME'
'MX'
'PTR'
'SRV'
'TXT'
```

For a "service_status" query item a valid value is one of:

```
'FAILED'
'INACTIVE'
'OFFLINE'
'WARNING'
'WORKING'
```

For a "zone_func" query item a valid value is one of:

```
'Forward-mapping Zone'
'IPv4 Reverse-mapping Zone'
'IPv6 Reverse-mapping Zone'
```

For a "zone_type" query item a valid value is one of:

```
'Authoritative'
'Delegation'.
'Forward'
'Stub'
```

For a "v_type" query item a valid value is one of:

```
'HostSystem'
'VirtualCenter'
'VirtualMachine'
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Getting value
my $value = $query_item->value();
# Modifying value
$query_item->value("Santa Clara");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, modify. This sample also includes error handling for the operations.

##Preparation prior to a smart folder query item object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Create a smart folder query item object

```
my $query_item = Infoblox::Grid::SmartFolder::QueryItem->new(
    name                    => "Country",
    is_extensible_attribute  => "true",
    operator                => "eq",
    value                   => "Canada",
);
unless ($query_item){
     die("Construct smart folder query item failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
}

print "SmartFolder::QueryItem object created successfully\n";
```

#Create a personal smart folder and use $query_item

```
#create a personal smart folder and use $query_item
my $smart_folder = Infoblox::Grid::PersonalSmartFolder->new(
   name         => "my_personal_folder",
   query_items  => [$query_item],
  );

#put smart folder into session
$session->add($smart_folder)
  or die("Smart folder creation failed: ",
          Infoblox::status_code() . ":" . Infoblox::status_detail());

print "Smart folder creation successful\n";
```

#Get and modify the query items

```
#get the personal smart folder object from session
my @result = $session->get(
    object => "Infoblox::Grid::PersonalSmartFolder",
    name   => "my_personal_folder"
);

unless (scalar(@result) == 0) {
    my $smart_folder = $result[0];

    if ($smart_folder) {

      #modify query_items
      $query_item->value("France");
      $smart_folder->query_items([$query_item]);

      #update session
      $session->modify($smart_folder)
        or die("modify smart_folder my_personal_folder failed: ",
            $session->status_code(), $session->status_detail());
          }
       } else {
        print "No smart folder found.";
       }
```

#Clean up

```
@result = $session->get(
    object => "Infoblox::Grid::PersonalSmartFolder",
    name   => "my_personal_folder"
);

unless (scalar(@result) == 0) {
    $session->remove($result[0]);
}

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Grid::GlobalSmartFolder, Infoblox::Grid::PersonalSmartFolder

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SmartFolderChildren - Smart Folder retrieval item.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->search( )
- METHODS
  - is_matching_object( )
  - value( )
  - value_type( )
  - comment( )
  - extensible_attributes( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SmartFolderChildren - Smart Folder retrieval item.

DESCRIPTION

This module helps read the objects that are associated with either a smart folder (global or personal) or a set of queries that users can make without saving a smart folder object on the appliance.

This interface can be used for both "filtering" and "grouping" the results of smart folder associated objects.

CONSTRUCTOR

The object cannot be constructed. It can only be returned as part of a search.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a personal smart folder object.

Infoblox::Session->search( )

Use this method to search for SmartFolderChildren objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
 Apply the following attributes to search for a specific SmartFolderChildren object:

  query_items                  - Required. A list of Infoblox::Grid::SmartFolder::QueryItem objects.
  smart_folder_name            - Required. PersonalSmartFolder/GlobalSmartFolder name in string
format.
  smart_folder_type            - Optional. "Global" / "Personal", "Global" if not specified.
  smart_folder_group_bys       - Optional. A list of Infoblox::Grid::SmartFolder::GroupBy objects.
  smart_folder_group_by_values - Optional. A list of Infoblox::Grid::SmartFolder::GroupByValue
objects.
  enable_no_values             - Optional. "true" | "false". "false" will not return "No Value" node
(child) unless explicitly enabled during the addition of the smart folder.

 Both "smart_folder_name" and "query_items" are marked "Required" but they cannot be specified
together. Note the following guidelines:

 - "query_items" cannot be specified when "smart_folder_name" is specified because if
"smart_folder_name" is specified, all the properties of a query can be derived from that smart
folder.

 - "query_items" can be specified with "smart_folder_group_bys". In this case, the results are
grouped by specified group_bys attribute and a list of intermediate objects are returned.

 - When only "smart_folder_name" is specified, it runs like a query using all properties from this
folder and returning the results accordingly.

 - When "smart_folder_group_bys" is specified with "smart_folder_name", the query is used from the
specified smart folder; however, the results are shown from the level listed in
"smart_folder_group_bys".
```

- "smart_folder_group_bys" by itself cannot be an argument since it does not provide sufficient input.

- "smart_folder_group_by_values" by itself cannot be an argument since it does not provide sufficient input.

- For the non-saved smart folder query, the "smart_folder_group_by_values" can be used to specify the level of results to return. "smart_folder_group_bys" must also be specified in this case.

Example

Please see the 'SAMPLE CODE' section below.

METHODS

This section describes all the methods that you can use to retrieve the attribute values of a SmartFolderChildren object.

is_matching_object( )

Use this method to retrieve the is_matching_object flag.

Parameter

The flag will be "true" if a matching object is returned inside the value(); if not, the attribute `value()` will contain the name of the intermediate object.

value( )

Use this method to retrieve the value of the object.

Parameter

This attribute will contain either an actual object matching the query (if is_matching_object is "true") or a string describing the name of the intermediate object.

Intermediate objects are the extensible attribute value objects for which the grouping was requested. Matching objects are the DB objects that matched the query specified.

value_type( )

Use this method to retrieve the type of the returned value.

Parameter

If is_matching_object is "false", this will contain one of: "STRING" / "INTEGER" / "DATE" / "BOOLEAN".

comment( )

Use this method to retrieve the comment.

Parameter

This attribute will return the value of the `comment()` attribute of the associated object, if `is_matching_object()` is "true". It will return undef if is_matching_object is "false".

extensible_attributes( )

Use this method to retrieve the extensible attributes.

Parameter

This attribute will return the value of the `extensible_attributes()` attribute of the associated object (if `is_matching_object()` is "true"). It will return undef if is_matching_object is "false".

SAMPLE CODE

```
USE-CASES & EXAMPLES:

##########################################################################
# Case 1 : Fetching objects based on the query defined for an existing folder
##########################################################################

my @retrieved_objs = $session->search(
            object =>"Infoblox::Grid::SmartFolderChildren" ,
```

```
            smart_folder_name => "global_smart_folder_1"
         );

my $object = $retrieved_objs[0];

if ($object->is_matching_object())
{
   print $object->comment();
   # $object->extensible_attributes() will contain the object's
   # extensible attributes
   #
   # This will be the actual matching object
   # $object->value();
}
else
{
   # Name of the intermediate group, use this to fetch another level if needed
   # This will be "USA", "Canada" etc
   # if the grouping was on "Country" for example
   print $object->value();
   print $object->value_type();
}

##############################################################################
# Case 2 : Fetching objects based on the smart folder and a given group_by_values
##############################################################################

my $groupbyvalue1 = Infoblox::Grid::SmartFolder::GroupByValue->new(
         name => "Country",
         value => "USA",
);
my $groupbyvalue2 = Infoblox::Grid::SmartFolder::GroupByValue->new(
         name => "State",
         value => "CA",
);

my  @retrieved_objs = $session->search(
      object =>"Infoblox::Grid::SmartFolderChildren" ,
      smart_folder_name => "global_smart_folder_1",
      smart_folder_group_by_values => [$groupbyvalue1, $groupbyvalue2],
         );

my $object = $retrieved_objs[0];
if ($object->is_matching_object())
{
   print $object->comment();
   # $object->extensible_attributes() will contain the object's
   # extensible attributes
   #
   # This will be the actual matching object
   # $object->value();
}
else
{
   # Name of the intermediate group, use this to fetch another level if needed
   print $object->value();
   print $object->value_type();
}

##############################################################################
# Case 3 : Fetching objects based on the not-saved smart folder query, no grouping
##############################################################################

my  @retrieved_objs = $session->search(
      object =>"Infoblox::Grid::SmartFolderChildren" ,
      #This query will only return intermediate objects
      query_items => [queryitem1, queryitem2],
         );

my $object = $retrieved_objs[0];
if ($object->is_matching_object())
{
   print $object->comment();
   # $object->extensible_attributes() will contain the object's
   # extensible attributes
   #
   # This will be the actual matching object
   # $object->value();

}
else
{
   # This shall never happen
}
```

```
##########################################################################
# Case 4 : Fetching objects based on the not-saved smart folder query & with top level grouping
##########################################################################

my  @retrieved_objs = $session->search(
      object =>"Infoblox::Grid::SmartFolderChildren" ,
      query_items => [queryitem1, queryitem2],
      smart_folder_group_bys => [$groupby1, $groupby2],
         );

my $object = $retrieved_objs[0];
if ($object->is_matching_object())
{
   # This shall never happen
}
else
{
   # Name of the intermediate group, use this to fetch another level if needed
   print $object->value();
   print $object->value_type();
}

##########################################################################
# Case 5 : Fetching objects based on the not-saved smart folder query & with second level grouping
##########################################################################

my  @retrieved_objs = $session->search(
      object =>"Infoblox::Grid::SmartFolderChildren" ,
      query_items => [$queryitem1, $queryitem2],
      smart_folder_group_bys => [$groupby1, $groupby2],
      smart_folder_group_by_values => [$groupbyvalue1],
         );

my $object = $retrieved_objs[0];
if ($object->is_matching_object())
{
   # This shall never happen
}
else
{
   # Name of the intermediate group, use this to fetch another level if needed
   # The intermediate objects grouped by $groupby2 are returned in this case
   print $object->value();
   print $object->value_type();
}

##########################################################################
# Case 6 : Fetching objects based on the defined SF query & grouping by a given rule "value"
##########################################################################

my $groupbyvalue1 = Infoblox::Grid::SmartFolder::GroupByValue->new(
        name => "Country",
        value => "USA",
);

my  @retrieved_objs = $session->search(
      object =>"Infoblox::Grid::SmartFolderChildren" ,
      smart_folder_name => "global_smart_folder_1"
      smart_folder_group_by_values => [$groupbyvalue1],
   );

my $object = $retrieved_objs[0];

if ($object->is_matching_object())
{
   print $object->comment();
   # $object->extensible_attributes() will contain the object's
   # extensible attributes
   #
   # This will be the actual matching object
   # $object->value();
}
else
{
   # Name of the intermediate group, use this to fetch another level if needed
   print $object->value();
   print $object->value_type();
}

##########################################################################
# Case 7 : Fetching next levels of data from result of previous one (NESTED example)
##########################################################################
```

```
# EXAMPLE: Get objects for Country USA-> State CA-> DB Objects.

#Add Personal Smart Folder with 2 levels of grouping
#Use this smart folder to fetch intermediate objects from first level (groups)
#Use the result of this first level and fetch second level of intermediate objects (groups)
#Use the result of this second level and fetch the final / third level of DB protocol objects

my $groupby1 = Infoblox::Grid::SmartFolder::GroupBy->new(
            enable_grouping => True,
            value => "Country",
            value_type => "EXTATTR",
);
my $groupby2 = Infoblox::Grid::SmartFolder::GroupBy->new(
            enable_grouping => True,
            value => "State",
            value_type => "EXTATTR",
);

my  $smartfolder = Infoblox::Grid::PersonalSmartFolder->new(
             name => "personal_smart_folder_1",
             comment => "personal smart folder 1 comment",
             enable_grouping => True,
             group_bys => [$groupby1, $groupby2],
             query_items => [],
              );
my $response = $session->add($smartfolder)

my  @retrieved_objs = $session->search(
      object =>"Infoblox::Grid::SmartFolderChildren" ,
      smart_folder_name => "personal_smart_folder_1",
      smart_folder_type => "Personal",
   );

my $object = $retrieved_objs[0];

if ($object->is_matching_object())
{
        # This will not happen since intermediate objects are returned in this case
}
else
{
   # Name of the intermediate group, use this to fetch another level if needed
   my $intermediate_group_value = $object->value();
        if ($intermediate_group_value eq "USA")
        {
            my $groupbyvalue_usa = Infoblox::Grid::SmartFolder::GroupByValue->new(
                name => "Country",
                value => "USA",
            );
            my  @retrieved_objs = $session->search(
                object =>"Infoblox::Grid::SmartFolderChildren" ,
                smart_folder_name => "personal_smart_folder_1"
                smart_folder_group_by_values => [$groupbyvalue_usa],
            );

            my $object = $retrieved_objs[0];

            my $intermediate2_group_value = $object->value();
            if ($intermediate2_group_value eq "CA")
            {
                  my $groupbyvalue_ca = Infoblox::Grid::SmartFolder::GroupByValue->new(
                      name => "State",
                      value => "CA",
                  );
                  my  @retrieved_objs = $session->search(
                      object =>"Infoblox::Grid::SmartFolderChildren" ,
                      smart_folder_name => "personal_smart_folder_1"
                      smart_folder_type => "Personal",
                      smart_folder_group_by_values => [$groupbyvalue_ca],
                  );

                  my $object = $retrieved_objs[0];

                   if ($object->comment() eq "I am a network object")
                   {
                       # $object here represents the Database object
                   }
            }
```

```
}
####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::PersonalSmartFolder, Infoblox::Session->add(), Infoblox::Session->fetch_smart_folder_objects(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(),Infoblox::Session

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SNMP::Admin - Manages a SNMP Admin object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid->snmp_admin( )
  - Infoblox::Grid::Member->snmp_admin( )
- METHODS
  - sysContact( )
  - sysContact_node1( )
  - sysContact_node2( )
  - sysDescr( )
  - sysDescr_node1( )
  - sysDescr_node2( )
  - sysLocation( )
  - sysLocation_node1( )
  - sysLocation_node2( )
  - sysName( )
  - sysName_node1( )
  - sysName_node2( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SNMP::Admin - Manages a SNMP Admin object.

DESCRIPTION

SNMP (Simple Network Management Protocol) is a protocol that administrators use to manage network devices and monitor their processes. An SNMP-managed appliance, such as an Infoblox appliance, has an SNMP agent that collects data and stores them as objects in MIBs (Management Information Bases). You can configure an Infoblox appliance as an SNMP-managed appliance. Infoblox appliances support SNMP versions 1, 2, and 3. You can enter values for sysContact, sysLocation, sysName, sysDescr managed objects in MIB-II, the standard MIB defined in RFC 1213. Once these values are set on the appliance, the administrators can send queries for these values from management systems that are allowed to send queries to the appliance.

CONSTRUCTOR

For SNMP Admin objects used as part of an HA pair

```
my $snmp = Infoblox::Grid::SNMP::Admin->new(
    sysContact_node1  => $string,      #Optional / Default is empty
    sysDescr_node1    => $string,      #Optional / Default is empty
    sysLocation_node1 => $string,  #Optional / Default is empty
    sysName_node1     => $string,  #Optional / Default is hostname of appliance
    sysContact_node2  => $string,      #Optional / Default is empty
    sysDescr_node2    => $string,      #Optional / Default is empty
    sysLocation_node2 => $string,  #Optional / Default is empty
    sysName_node2     => $string,  #Optional / Default is hostname of appliance
 );
```

otherwise

```
my $snmp = Infoblox::Grid::SNMP::Admin->new(
    sysContact  => $string,  #Optional / Default is empty
    sysDescr    => $string,  #Optional / Default is empty
    sysLocation => $string,  #Optional / Default is empty
    sysName     => $string,  #Optional / Default is hostname of appliance
 );
```

When the SNMP Admin object is not used as part of an HA pair, sysContact/sysContact_node1, sysDescr/sysDescr_node1, ... are equivalent. The _node2 fields will be empty.

MODULE METHODS

The following functions are available to apply to an SNMP admin object.

Infoblox::Grid->snmp_admin( )

Use this function to specify an SNMP admin at the grid level. See Infoblox::Grid->snmp_admin() for parameters and return values.

Example

```
#Construct an object
 my $snmp = Infoblox::Grid::SNMP::Admin->new(
   sysContact  => "Somebody",
   sysDescr    => "SunOS noc1 5.6 Generic_105181-03 sun4u",
   sysLocation => "Infoblox Headquarter",
   sysName     => "noc1"
);

#Configure snmp at grid level
my $response = $grid->snmp_admin($snmp);
```

Infoblox::Grid::Member->snmp_admin( )

Use this function to specify an SNMP admin at the member level. See Infoblox::Grid::Member->snmp_admin() for parameters and return values.

Example

```
#Construct an object
  my $snmp = Infoblox::Grid::SNMP::Admin->new(
    sysContact_node1  => "Somebody",
    sysDescr_node1    => "SunOS noc1 5.6 Generic_105181-03 sun4u",
    sysLocation_node1 => "Infoblox Headquarter",
    sysName_node1     => "noc1"
);

# Configure snmp at member level
$member->snmp_admin($snmp);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an SNMP admin object.

sysContact( )

Use this method to set or retrieve the name of the contact person for the Simple Network Management Protocol (SNMP) agent host.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the contact person in string format. The maximum length of the name is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysContact
my $sysContact = $snmp->sysContact();
#Modify sysContact
$snmp->sysContact("Infoblox");
```

sysContact_node1( )

Use this method to set or retrieve the name of the contact person for the Simple Network Management Protocol (SNMP) agent host on node 1 of an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the contact person in string format. The maximum length of the name is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysContact_node1
my $sysContact_node1 = $snmp->sysContact_node1();
#Modify sysContact_node1
$snmp->sysContact_node1("Infoblox");
```

sysContact_node2( )

Use this method to set or retrieve the name of the contact person for the Simple Network Management Protocol (SNMP) agent host on node 2 of an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The name of the contact person in string format. The maximum length of the name is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysContact_node2
my $sysContact_node2 = $snmp->sysContact_node2();
#Modify sysContact_node2
$snmp->sysContact_node2("Infoblox");
```

sysDescr( )

Use this method to set or retrieve the name and version ID of the system's hardware type, the software Operational System (OS), and networking software that the appliance is running.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Useful information about the hardware and software described in string format. The maximum length of the sysDescr is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysDescr
my $sysDescr = $snmp->sysDescr();
#Modify sysDescr
$snmp->sysDescr("SunOS noc2 5.6 Generic_105181-03 sun5u");
```

sysDescr_node1( )

Use this method to set or retrieve the name and version ID of the system's hardware type, the software Operational System (OS), and networking software that the appliance is running on node 1 of an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Useful information about the hardware and software described in string format. The maximum length of the sysDescr_node1 is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysDescr_node1
my $sysDescr_node1 = $snmp->sysDescr_node1();
#Modify sysDescr_node1
$snmp->sysDescr_node1("SunOS noc2 5.6 Generic_105181-03 sun5u");
```

sysDescr_node2( )

Use this method to set or retrieve the name and version ID of the system's hardware type, the software Operational System (OS), and networking software that the appliance is running on node 2 of an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Useful information about the hardware and software described in string format. The maximum length of the sysDescr_node2 is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysDescr_node2
my $sysDescr_node2 = $snmp->sysDescr_node2();
#Modify sysDescr_node2
$snmp->sysDescr_node2("SunOS noc2 5.6 Generic_105181-03 sun5u");
```

sysLocation( )

Use this method to set or retrieve the physical location of the appliance.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The physical location of the appliance in string format. The maximum length of the sysLocation is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysLocation
my $sysLocation = $snmp->sysLocation();
#Modify sysLocation
$snmp->sysLocation("Infoblox Headquarter5");
```

sysLocation_node1( )

Use this method to set or retrieve the physical location of the appliance on node 1 of an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The physical location of the appliance in string format. The maximum length of the sysLocation_node1 is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysLocation_node1
my $sysLocation_node1 = $snmp->sysLocation_node1();
#Modify sysLocation_node1
$snmp->sysLocation_node1("Infoblox Headquarter5");
```

sysLocation_node2( )

Use this method to set or retrieve the physical location of the appliance on node 2 of an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The physical location of the appliance in string format. The maximum length of the sysLocation_node2 is 50 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails. If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysLocation_node2
my $sysLocation_node2 = $snmp->sysLocation_node2();
#Modify sysLocation_node2
$snmp->sysLocation_node2("Infoblox Headquarter5");
```

sysName( )

Use this method to set or retrieve the fully qualified domain name (FQDN) of the appliance.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The fully qualified domain name (FQDN) of the appliance in string format. The maximum length of the sysName is 50 characters. The default value of sysName is the hostname of the Infoblox appliance.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysName
my $sysName = $snmp->sysName();
#Modify sysName
$snmp->sysName("Infobloxnoc2");
```

sysName_node1( )

Use this method to set or retrieve the fully qualified domain name (FQDN) of the appliance on node 1 of an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The fully qualified domain name (FQDN) of the appliance in string format. The maximum length of the sysName_node1 is 50 characters. The default value of sysName_node1 is the hostname of the Infoblox appliance.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysName_node1
my $sysName_node1 = $snmp->sysName_node1();
#Modify sysName_node1
$snmp->sysName_node1("Infobloxnoc2");
```

sysName_node2( )

Use this method to set or retrieve the fully qualified domain name (FQDN) of the appliance on node 2 of an HA pair.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The fully qualified domain name (FQDN) of the appliance in string format. The maximum length of the sysName_node2 is 50 characters. The default value of sysName_node2 is the hostname of the Infoblox appliance.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get sysName_node2
my $sysName_node2 = $snmp->sysName_node2();
#Modify sysName_node2
$snmp->sysName_node2("Infobloxnoc2");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object such as add, modify and remove. This sample also includes error handling for the operations.

#Preparation prior to an SNMP admin object insertion

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Creating an SNMP admin object

```
my $snmp = Infoblox::Grid::SNMP::Admin->new(
    sysContact  => "Somebody",
    sysDescr    => "SunOS noc1 5.6 Generic_105181-03 sun4u",
    sysLocation => "Infoblox Headquarter",
    sysName     => "noc1"
    );
unless($snmp) {
     die("Construct a SNMP Admin failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "SNMP Admin object created successfully\n";
```

#Getting the grid object from the appliance through session

```
 my @result = $session->get(
    object => "Infoblox::Grid",
    name   => "Infoblox"
    );

my $grid = $result[0];
unless ($grid) {
       die("Get Grid object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get Grid object found at least 1 matching entry\n";

      #Modifying the value of the specified object.
        $grid->snmp_admin($snmp);  #Applying the changes.
        $session->modify($grid);
        print "SNMP Admin modified successfully\n";
```

#Modify SNMP admin attribute sysName

```
 #Modify existing attribute sysName
 $snmp->sysName("noc2.infoblox.com");

 #Apply changes to the Grid object.
 $grid->snmp_admin($snmp);

 #Update Grid object through the Infoblox session.
 $session->modify($grid)
```

```
            or die("modify Grid object failed: ",
                $session->status_code() . ":" . $session->status_detail());
        print "Modified SNMP Admin sysName of Grid object successful\n";
```

#Remove SNMP admin attribute from Grid object

```
  #Apply changes to the Grid object to Un-def SNMP Admin object.
  $grid->snmp_admin(undef);

  #Update Grid object through the Infoblox session.
  $session->modify($grid)
            or die("Remove SNMP Admin from Grid object failed: ",
                $session->status_code() . ":" . $session->status_detail());
        print "Removed SNMP Admin from Grid object successful\n";

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Grid

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SNMP::QueriesUser - Grid SNMP QueriesUser object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- METHODS
  - user()
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SNMP::QueriesUser - Grid SNMP QueriesUser object.

DESCRIPTION

The Grid SNMP Queriesuser object is used to set or retrieve the user name of an SNMPv3 user.

CONSTRUCTOR

```
my $query_user1 = Infoblox::Grid::SNMP::QueriesUser->new(
        user => $snmp_user1,
        comment => "query comment 1"
);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an SNMPv3 QueriesUser object.

user()

Use this method to set or retrieve the user name of an SNMPv3 user.

Parameter

A reference to the Infoblox::Grid::SNMP::User object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get query user
my $query_user=$grid->snmpv3_query_users();
#Set query user
$grid->snmpv3_query_users([$query_user1,$query_user2];
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::SNMP::User

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SNMP::TrapReceiver - Grid SNMP TrapReceiver object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- METHODS
  - address()
  - user()
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SNMP::TrapReceiver - Grid SNMP TrapReceiver object.

DESCRIPTION

The Grid SNMP TrapReceiver object is used to set or retrieve the IP address and user name of a trap receiver.

CONSTRUCTOR

```
my $Infoblox::Grid::SNMP::TrapReceiver->new(
    address => $ip_address,  #Required
    user => $snmpv3user      #Required
);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an SNMPv3 TrapReceiver object.

address()

Use this method to set or retrieve the IP address of the trap receiver.

Parameter

IPv4 address of the trap receiver

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get trap receiver address
my $ip=$trap_receiver->address();
#Set trap receiver address
$trap_receiver->address("192.168.1.2");
```

user()

Use this method to set or retrieve the user name for the trap receiver.

Parameter

A reference to the Infoblox::Grid::SNMP::User object or the user name in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get trap receiver user
my $user=$trap_receiver->user();
#Set trap receiver user
```

```
$trap_receiver->user($user);
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::SNMP::User

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SNMP::User - Grid SNMP User object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->search( )
  - Infoblox::Session->remove( )
- METHODS
  - auth_password( )
  - auth_protocol( )
  - comment( )
  - disable( )
  - engine_id( )
  - name( )
  - privacy_password( )
  - privacy_protocol( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SNMP::User - Grid SNMP User object

DESCRIPTION

The Grid SNMP User object is used to set, get, search, modify, or remove an SNMPv3 user. An SNMPv3 user is a managment system that sends queries and receives traps from the appliance.

CONSTRUCTOR

```
#Constructor for a Grid SNMP User object
my $snmp_user = Infoblox::Grid::SNMP::User->new(
        name                => $string,                 #Required
        auth_protocol       => "none"| "md5" | "sha",   #Required
        privacy_protocol    => "none"| "des" | "aes",   #Required
        auth_password       => $string,                 #Optional. Required for MD5/SHA authentication
        comment             => $string,                 #Optional / Default is undefined
        disable             => "true" | "false",        #Optional / Default is "false"
        privacy_password    => $string,                 #Optional. Required for MD5/SHA authentication
 );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a Grid SNMP User object.

Infoblox::Session->get( )

Use this method to retrieve existing objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get a Grid SNMP User object:

name     -  user name in string format
comment  -  comment string
```

Example

```
my @retrieved_objs = $session->get(
        object      => "Infoblox::Grid::SNMP::User",
        name        => "testuser" );
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use this method to modify the comment
$snmp_user->comment("This is a modified comment");
# Submit modification
my $response = $session->modify( $snmp_user );
```

Infoblox::Session->search( )

Use this method to search for Grid SNMP User objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a Grid SNMP User object:

 name -  user name (regular expression)
 comment - comment string (regular expression)
```

Example

```
# Search for all Grid SNMP User objects that start with "test"
my @retrieved_objs = $session->search(
    object      => "Infoblox::Grid::SNMP::User",
    name        => "test.*",
);
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the object, and then submit it for removal.

Example

```
# Get Grid SNMP User objects with the same starting name
my @retrieved_objs = $session->get(
    object      => "Infoblox::Grid::SNMP::User",
    name        => "testuser" );
# find the desired object from the retrieved list.
my $desired_user = $retrieved_objs[0];
# Submit for removal
my $response = $session->remove( $desired_user );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an SNMPv3 user.

auth_password( )

Use this method to provide an authentication password for the user.

Parameter

Authentication password of the user in string format.

Returns

If a parameter is specified, the method returns true when the modification succeeds, and returns false when the operation fails.

If parameter is not specified, the method returns the attribute value.

Example

```
#Get auth_password
my $password = $snmp_user->auth_password();
#Set the password
$snmp_user->auth_password("mysecret");
```

auth_protocol( )

Use this method to specify the authentication protocol of the user.

Parameter

Possible values for the authentication protocol are:

```
none
md5
sha
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get auth_protocol
my $proto = $snmp_user->auth_protocol();
#Set the protocol
$snmp_user->auth_protocol("md5");
```

comment( )

Use this method to provide a comment for the SNMPv3 User object.

Parameter

Comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get comment
my $comment = $snmp_user->comment();
#Set the comment
$snmp_user->comment("This is a sample comment");
```

disable( )

Use this method to configure or retrieve the "disable" flag.

The default value is false; that is, the user is enabled.

Parameter

Specify "true" to set the disable flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable flag
my $disabled = $snmp_user->disable();
#Set the disable flag
$snmp_user->disable("true");
```

engine_id( )

Use this method to retrieve the SNMP engine ID.

This field is read-only.

Parameter

None

Returns

When called without parameters, the method returns the attribute value. Calling this method with a parameter results in an error.

Example

```
#Get engine_id
my $engine_id = $snmp_user->engine_id();
```

name( )

Use this method to set or retrieve the name of an SNMPv3 user.

Parameter

Text with the name of the user.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get name
my $name = $snmp_user->name();
#Modify name
$snmp_user->name("test_user");
```

privacy_password( )

Use this method to provide a password for the privacy protocol.

Parameter

Password in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get privacy_password
my $password = $snmp_user->privacy_password();
#Set the password
$snmp_user->privacy_password("mysecret");
```

privacy_protocol( )

Use this method to specify encryption protocol for the SNMP packets.

Parameter

Possible values for the encryption protocol are:

```
none
des
aes
```

The values are case insensitive.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get privacy_protocol
my $proto = $snmp_user->privacy_protocol();
#Set the protocol
$snmp_user->privacy_protocol("aes");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Grid::SNMP::TrapReceiver, Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::SyslogServer - Syslog server object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid->syslog_server( )
  - Infoblox::Grid::Member->syslog_server( )
- METHODS
  - connection_type( )
  - ipv4addr( )
  - local_interface( )
  - message_source( )
  - port( )
  - severity( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::SyslogServer - Syslog server object.

DESCRIPTION

Syslog is a widely used mechanism for logging system events. Infoblox appliances generate syslog messages which can be viewed through the system log viewer and downloaded to a directory on your management station. In addition, the Infoblox appliance can be configured to send the messages to one or more external syslog servers for later analysis. Syslog messages provide information about appliance operations and processes. Also, the appliance can include audit log messages and specific BIND messages among the messages sent to the syslog server.

CONSTRUCTOR

```
 my $syslog_server = Infoblox::Grid::SyslogServer->new(
        ipv4addr        => $ipv4addr,                        #Required
        connection_type => "tcp" | "udp",                     #Optional / Default is "udp"
        local_interface => "any" | "mgmt" | "lan",            #Optional / Default is "any"
        message_source  => "any" | "internal" | "external",   #Optional / Default is "any"
        port            => $num,                             #Optional / Default is 514
        severity        => "debug" | "info" | "notice" | "warning" | "err" | "crit" | "alert" | "emerg",
#Optional / Default is "debug"
 );
```

MODULE METHODS

The following functions are available to be applied to a syslog server object.

Infoblox::Grid->syslog_server( )

Use this function to specify a syslog server at the grid level on the Infoblox appliance. See Infoblox::Grid->syslog_server() for parameters and return values.

Example

```
#Construct a syslog server object
my $syslog_server1 = Infoblox::Grid::SyslogServer->new(
                        ipv4addr => "1.1.1.1",
                       );

#Configure syslog server on the Infoblox grid object
my $responce = $grid->syslog_server([$syslog_server1]);
```

Infoblox::Grid::Member->syslog_server( )

Use this function to specify a syslog server at the member level on the Infoblox appliance. See Infoblox::Grid::Member->syslog_server() for parameters and return values.

Example

```
#Construct a syslog server object
my $syslog_server2 = Infoblox::Grid::SyslogServer->new(
                       ipv4addr => "2.2.2.2",
                       severity => "debug",
                     );

#Configure syslog server on the grid member object
my $responce = $grid_member->syslog_server([$syslog_server2]);
```

METHODS

This section describes all the methods that can be used to configure and retrieve the attribute value of a syslog server object.

connection_type( )

Use this method to set or retrieve the connection type to the external syslog server. This method specifies whether the appliance uses TCP or UDP to connect to the external syslog server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are: `"tcp"`, `"udp"`. The default value is "udp".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#get connection_type
my $connection_type = $syslog_server->connection_type();
#Modify connection_type
$syslog_server->connection_type("tcp");
```

ipv4addr( )

Use this method to set or retrieve the IP address of a syslog server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

An IPv4 address is a 32-bit number in dotted decimal notation. It consists of four 8-bit groups of decimal digits separated by decimal points (example: 192.168.1.2).

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#get ipv4addr
my $ipv4addr = $syslog_server->ipv4addr();
#Modify ipv4addr
$syslog_server->ipv4addr("3.3.3.3");
```

local_interface( )

Use this method to set or retrieve the interface through which the appliance sends syslog messages to the syslog server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are: `"any"`, `"mgmt"`, `"lan"`. The default value is "any".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#get local_interface
my $local_interface = $syslog_server->local_interface();
#Modify local_interface
$syslog_server->local_interface("lan");
```

message_source( )

Use this method to set or retrieve the message source value of the syslog server. This method specifies which syslog messages the appliance sends to the external syslog server:

```
 Any     : The appliance sends both internal and external syslog messages.
 Internal: The appliance sends syslog messages that it generates.
 External: The appliance sends syslog messages that it receives from other appliances, such as syslog
servers and routers.
```

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are: `"any"`, `"internal"`, `"external"`. The default value is "any".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#get message_source
my $message_source = $syslog_server->message_source();
#Modify message_source
$syslog_server->message_source("internal");
```

port( )

Use this method to set or retrieve the destination port number.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are between 1 and 65535. The default value is 514.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#get port
my $port = $syslog_server->port();
#Modify port
$syslog_server->port("515");
```

severity( )

Use this method to set or retrieve the severity level for the syslog server. When you set a severity level, grid members send messages for that severity level plus all messages for all severity levels above it. The lowest severity level is "debug", and the highest severity level is "emerg".

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are: `"debug"`, `"info"`, `"notice"`, `"warning"`, `"err"`, `"crit"`, `"alert"`, `"emerg"`. The default value is "debug".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#get severity
my $severity = $syslog_server->severity();
#Modify severity
$syslog_server->severity("info");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to a syslog server object such as add, modify, and remove. Also, this sample code includes error handling for the operations.

#Preparation prior to a syslog server object insertion

```
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",   #appliance host ip
    username => "admin",         #appliance user login
    password => "infoblox"       #appliance password
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully.\n";
```

#Create a syslog server object

```
my $syslog_server = Infoblox::Grid::SyslogServer->new(
                    ipv4addr => "1.1.1.1",
                    severity => "debug"
                   );

unless($syslog_server) {
       die("Construct syslog server object failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "syslog server object created successfully\n";
```

#Get Infoblox grid object and add syslog server to it

```
#Get Infoblox grid object
my @grid_objs = $session->get(
           object => "Infoblox::Grid",
           name   => "Infoblox"
       );

my $grid = $grid_objs[0];
unless ($grid) {
       die("Get Grid object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get Grid object found at least 1 matching entry\n";

#Apply the changes to the Grid object
$grid->syslog_server([$syslog_server]);

#Apply the changes
$session->modify($grid)
    or die("Modify Grid object failed: ",
            $session->status_code() . ":" . $session->status_detail());

print "Grid object with syslog server updated to Infoblox appliance successfully\n";
```

#Modify syslog server object

```
#Get Infoblox grid object
my @grid_objs = $session->get(
           object => "Infoblox::Grid",
           name   => "Infoblox"
       );

unless(scalar(@grid_objs) == 0){
       my $grid = $grid_objs[0];
       if($grid){
```

```
                my $syslog_obj = $grid->syslog_server();
                my @sys_servers = @{$syslog_obj};
                my $sys_server = $sys_servers[0];

                #Modifying the ipv4addr of the syslog server object.
                $sys_server->ipv4addr("2.2.2.2");

                #Modifying the connection_type of the syslog server object.
                $sys_server->connection_type("tcp");

                #Modifying the local_interface of the syslog server object.
                $sys_server->local_interface("lan");

                #Modifying the message_source of the syslog server object.
                $sys_server->message_source("external");

                #Modifying the port of the syslog server object.
                $sys_server->port("516");

                #Modifying the severity of the syslog server object.
                $sys_server->severity("warning");

                #Apply changes to the Grid object.
                $grid->syslog_server([$sys_server]);

                #Update Grid object through the Infoblox session.
                $session->modify($grid)
                            or die("modify Grid object failed: ",
                                $session->status_code() . ":" . $session->status_detail());

                print "Grid object with modified syslog server updated to Infoblox appliance
successfully\n";
                        }
                 }
```

#Remove syslog server

```
 #Get Infoblox grid object
 my @grid_objs = $session->get(
            object => "Infoblox::Grid",
            name   => "Infoblox"
        );

 my $grid = $grid_objs[0];
 unless ($grid) {
        die("Get Grid object failed: ",
             $session->status_code() . ":" . $session->status_detail());
 }
 print "Get Grid object found at least 1 matching entry\n";

 #Remove syslog servers from the Grid object.
 $grid->syslog_server(undef);

 #Update Grid object through the Infoblox session.
 $session->modify($grid)
             or die("Remove syslog server from Grid object failed: ",
                 $session->status_code() . ":" . $session->status_detail());

 print "Removed syslog server from Grid object successfully\n";

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Grid, Infoblox::Grid::Member

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Grid::TimeZone - Grid time zone object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - ntp_enabled( )
  - time_zone( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Grid::TimeZone - Grid time zone object.

DESCRIPTION

Grid TimeZone object is used to set the time zone for an Infoblox appliance.

CONSTRUCTOR

```
#Constructor for an Grid Time zone object
my $time_zone = Infoblox::Grid::TimeZone->new (
        time_zone    => $time_zone            #Required
        ntp_enabled  => "true" | "false"      #Optional / Default "false"
    );
```

SESSION METHODS

This section describes all the methods in Infoblox::Session that you can apply to a grid TimeZone object.

Infoblox::Session->get( )

Use this method to retrieve existing objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get a grid TimeZone object:

none.
```

Example

```
my @result_array = $session->get( "object" => "Infoblox::Grid::TimeZone");
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Use method to modify the time zone.
$object->time_zone( "(UTC - 8:00) Pacific Time (US and Canada), Tijuana" );
$object->ntp_enabled( "false" );
# Submit modification
my $response = $session->modify( $object );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute value of a time zone.

ntp_enabled( )

Use this method to enable/disable NTP (Network Time Protocol).

Parameter

Specify "true" to enable NTP or "false" to disable it. Default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get ntp_enabled
my $enable = $object->ntp_enabled();
#Modify ntp_enabled
$object->ntp_enabled("true");
```

time_zone( )

Use this method to set or retrieve a time_zone.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Desired time in string format. Following are the available Time Zones:

```
'(UTC - 12:00)'
'(UTC - 11:00) Midway Island, Samoa'
'(UTC - 10:00) Hawaii'
'(UTC - 9:00) Alaska'
'(UTC - 8:00) Pacific Time (US and Canada), Tijuana'
'(UTC - 7:00) Arizona'
'(UTC - 7:00) Mountain Time (US and Canada)'
'(UTC - 6:00) Central Time (US and Canada)'
'(UTC - 6:00) Mexico City, Tegucigalpa'
'(UTC - 6:00) Saskatchewan'
'(UTC - 5:00) Bogota, Lima, Quito'
'(UTC - 5:00) Eastern Time (US and Canada)'
'(UTC - 5:00) Indiana (East)'
'(UTC - 4:00) Atlantic Time (Canada)'
'(UTC - 4:00) Caracas'
'(UTC - 4:00) Santiago'
'(UTC - 3:30) Newfoundland'
'(UTC - 3:00) Brasilia'
'(UTC - 3:00) Buenos Aires'
'(UTC - 2:00) Mid-Atlantic'
'(UTC - 1:00) Azores'
'(UTC) Casablanca'
'(UTC) Coordinated Universal Time'
'(UTC) Dublin'
'(UTC) Lisbon'
'(UTC) London'
'(UTC + 1:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna'
'(UTC + 1:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague'
'(UTC + 1:00) Brussels, Copenhagen, Madrid, Paris, Vilnius'
'(UTC + 1:00) Sarajevo, Skopje, Sofija, Warsaw, Zagreb'
'(UTC + 2:00) Athens, Istanbul, Minsk'
'(UTC + 2:00) Bucharest'
'(UTC + 2:00) Cairo'
'(UTC + 2:00) Harare'
'(UTC + 2:00) Helsinki'
'(UTC + 2:00) Jerusalem'
'(UTC + 3:00) Baghdad, Kuwait, Riyadh'
'(UTC + 3:00) Moscow, St    .   Petersburg, Volgograd'
'(UTC + 3:00) Nairobi'
'(UTC + 3:30) Tehran'
'(UTC + 4:00) Dubai'
'(UTC + 4:00) Baku'
'(UTC + 4:00) Kabul'
'(UTC + 5:00) Ekaterinburg'
'(UTC + 5:00) Islamabad, Karachi'
'(UTC + 5:30) Bombay, Calcutta, Madras, New Delhi'
'(UTC + 6:00) Dhaka'
'(UTC + 6:00) Colombo'
'(UTC + 7:00) Bangkok, Hanoi'
'(UTC + 8:00) Beijing, Chongqing, Shanghai'
'(UTC + 8:00) Hong Kong'
'(UTC + 8:00) Perth'
'(UTC + 8:00) Singapore'
'(UTC + 8:00) Taipei'
'(UTC + 9:00) Osaka, Sapporo, Tokyo'
'(UTC + 9:00) Seoul'
'(UTC + 9:30) Adelaide'
'(UTC + 9:30) Darwin'
'(UTC + 10:00) Brisbane'
'(UTC + 10:00) Canberra, Sydney'
'(UTC + 10:00) Melbourne, Victoria'
'(UTC + 10:00) Guam'
'(UTC + 10:00) Hobart'
```

```
'(UTC + 10:00) Vladivostok'
'(UTC + 11:00) Magadan'
'(UTC + 11:00) Solomon Islands'
'(UTC + 12:00) Auckland'
'(UTC + 12:00) Fiji'
'(UTC + 12:00) Marshall Islands'
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#Get time_zone
my $time_zone = $object->time_zone();
#Modify time_zone
$object->time_zone("(UTC - 8:00) Pacific Time (US and Canada), Tijuana");
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to an object such as create, modify, and remove an object. This sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my @result_array = $session->get( "object" => "Infoblox::Grid::TimeZone");
if( defined @result_array  and scalar( @result_array ) > 0 )
 {
        my $object = $result_array[0];
        unless ($object) {
                die("Get on Time zone failed: ",
        $session->status_code() . ":" . $session->status_detail());
        }
#Modify time_zone
$object->time_zone("(UTC - 8:00) Pacific Time (US and Canada), Tijuana");
my $enable = $object->ntp_enabled();
$object->ntp_enabled( "true" );
$session->modify($object)
   or die("Modify on Time zone failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Grid Time zone modified successfully \n";
}
else
{
        print "No  TimeZone Structure found !!!\n";
}

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->get(), Infoblox::Session->modify()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::IFMap::AuthRule - IF-MAP Authorization Rules object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::Grid::IFMap->authorization_rules( )
  - Infoblox::IFMap::Client->authorization_rules( )
- METHODS
  - custom_identifier_type( )
  - identifier( )
  - metadata( )
  - operation( )
  - permission( )
  - publisher( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::AuthRule - IF-MAP Authorization Rules object.

DESCRIPTION

This object allows administrators to configure the access rights of IF-MAP clients. The authorization rules grant or deny access to a set of operations, and can be configured to match a specific identifier type and/or a specific metadata type.

CONSTRUCTOR

```
 my $authrule=Infoblox::IFMap::AuthRule->new(
              identifier => "ip-address" | "mac-address" | "device" | "identity" |
                            "access-request" | "custom" | "all",                  #Required
              operation  => "publish" | "subscribe" | "purge" | "search" | "all",    #Required
              permission => "grant" | "deny",                                       #Required
              custom_identifier_type => $cit,                                       #Optional / Default is
undef,
                                                                                   #Required only if
identifier is set to "custom".
              metadata   => $string,                                                #Optional / Default is
all metadata types
             );
```

MODULE METHODS

Infoblox::Grid::IFMap->authorization_rules( )

Use this method to set or retrieve the IF-MAP server authorization rules that apply to all IF-MAP clients. (See Infoblox::Grid::IFMap->authorization_rules( ))

Example

```
# Get authorization_rules
my $ref_authorization_rules = $gridifmap->authorization_rules( );

# Create an authorization rule allowing clients to publish Location on IP addresses
my $rule = Infoblox::IFMap::AuthRule->new(
     identifier => "ip-address",
     metadata => "Location"
     operation => "publish",
     permission => "grant"
     );

# Set authorization_rules
$gridifmap->authorization_rules( [$rule] );
```

Infoblox::IFMap::Client->authorization_rules( )

Use this method to set or retrieve the IF-MAP authorization rules defined for a particular IF-MAP client. (See Infoblox::IFMap::Client->authorization_rules( ))

Examples

```
# Get authorization_rules
my $ref_authorization_rules = $ifmapclient->authorization_rules( );

# Create an authorization rule preventing clients from publishing anything
my $rule = Infoblox::IFMap::AuthRule->new(
     operation  => "publish",
     identifier => "all",
     permission => "deny"
     );

# Set authorization_rules
$ifmapclient->authorization_rules( [$rule] );
```

METHODS

custom_identifier_type( )

Use this method to set or retrieve the value of the custom_identifier_type attribute. This attribute must be specified if the identifier value is set to "custom".

Parameter

The valid value of this parameter is a reference to the Infoblox::IFMap::Identifier::Definition object. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Get the authorization rule custom identifier type
my $auth_rule_CIT = $authrule->custom_identifier_type();

#Set the authorization rule custom identifier type
$authrule->custom_identifier_type($auth_rule_CIT);
```

identifier( )

Use this method to set or retrieve the value of the identifier attribute. The value must be a single identifier type or "all" to specify that this rule applies to all identifier types. Valid identifier types are: ip-address, mac-address, device, identity and access-request.

Parameter

The method returns or sets the identifier value. The valid values are the following:

```
"all"
"custom"
"ip-address"
"mac-address"
"device"
"identity"
"access-request"
```

Note that if the identifier is set to "custom", the custom_identifier_type field must be set.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get the identifier
my $identifier = $authrule->identifier( );

# Set the identifier to a single identifier type (IP address)
$authrule->identifier( "ip-address" );
```

metadata( )

Use this method to set or retrieve the value of the metadata attribute. The value can be a single expression of type or undef to specify that this rule applies to all metadata types. This is a free form expression that depends on the IF-MAP schema. Basic metadata types defined by the IF-MAP specifications include: ip-mac, access-request-ip, event, role, authenticated-as, etc.

Parameter

The parameter is a string that contains an expression conforming to the IF-MAP schema. Specify an empty string for all metadata.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get metadata
my $metadata = $authrule->metadata( );

# Set metadata to a single type (role)
$authrule->metadata( "role" );
```

operation( )

Use this method to set or retrieve the value of the operation attribute. The value can be a single operation or "all" to specify that this rule applies to all operations. Allowed values are: publish, subscribe, poll, purge, search and all.

Parameter

The method returns or sets the operation attribute. The following are valid values for this attribute:

```
"all"
"publish"
"subscribe"
"poll"
"purge"
"search"
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get operation
my $operation = $authrule->operation( );

# Set operation
$authrule->operation( "publish" );
```

permission( )

Use this method to set or retrieve the value of the permission attribute. Allowed values are: grant and deny.

Parameter

Allowed values for the permission parameter are

```
"grant"
"deny"
```

The default authorization rule contains 'deny' permission for all. By adding authorization rules to the rule chain, it is possible to change permissions for all clients in a grid or for individual clients.

Examples

```
# Get permission
my $permission = $authrule->permission( );

# Set permission
$authrule->permission( "grant" );
```

publisher( )

Use this method to set or retrieve the value of publisher attribute. This attribute applies to purge operations only.

Parameter

The allowed values for the publisher parameter are:

```
"self"
"all"
```

The value "self" indicates that this authorization rule to purge data applies only to the metadata published by the client that sent the request. The value "all" indicates that the rule applies to metadata published by all clients.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get permission
my $publisher = $authrule->publisher();

# Set permission
$authrule->publisher( "self" );
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as get and modify. This sample code also includes error handling for the operations.

#Preparations

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox  appliance
my $session = Infoblox::Session->new(
   master   => "192.168.1.2", #appliance host ip
   username => "admin",       #appliance user login
   password => "infoblox"     #appliance password
);

unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "Session created successfully\n";

#Create an IF-MAP AuthRule object

my $authrule = Infoblox::IFMap::AuthRule->new(
                      identifier => "ip-address",
                      metadata   => "meta:layer3-authorization",
                      operation  => "publish",
                      permission => "grant",
               );
unless($authrule) {
       die("Construct IF-MAP Authorization rule object failed: ",
            Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "Authorization rule object created successfully\n";
```

#Get the IF-MAP Server object and add an authorization rule object to it

```
#Get the System IF-MAP Object
my @retrieved_objs = $session->get(
    object => "Infoblox::Grid::IFMap",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get System IF-MAP Object failed: ",
            $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Get System IF-MAP Object found at least 1 matching entry\n";
```

#Apply the changes to the IF-MAP Server object

```
$object->authorization_rules([$authrule]);
```

```
#Apply the changes
$session->modify($object)
    or die("Modify System IF-MAP Object failed: ",
            $session->status_code( ) . ":" . $session->status_detail( ));
print "System IF-MAP Object with new authorization rule updated to Infoblox appliance successfully\n";
```

#Modify authorization rules

```
#Modifying the identifier of authorization rule object.
unless($authrule->identifier("mac-address")){
    die("Modification of the identifier field of authorization rule failed");
}

#Modifying the operation of authorization rule object.
unless($authrule->operation("search")){
    die("Modification of the operation field of authorization rule failed");
}
```

#Apply the changes to the IF-MAP Server object.

```
$object->authorization_rules([$authrule]);

#Update the IF-MAP Server object through the Infoblox session.
$session->modify($object)
            or die("modify System IF-MAP Object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
       print "System IF-MAP Object with authorization rule updated to Infoblox appliance successfully\n";
```

#Reset the authorization rules

```
#Reset the authorization rules to empty list
$object->authorization_rules([]);

#Update the IF-MAP Server object through the Infoblox session.
$session->modify($object)
            or die("Remove of the authorization rule from System IF-MAP Object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
       print "Removed the authorization rule from System IF-MAP Object successfully\n";
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::Identifier::Definition, Infoblox::Grid::IFMap, Infoblox::IFMap::Client, Infoblox::Grid::IFMap->authorization_rules( ), Infoblox::IFMap::Client->authorization_rules( )

COPYRIGHT

Copyright (c) 1998-2011 Infoblox Inc.

Infoblox::IFMap::CACertificate - IF-MAP CA Certificate object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - certificate( )
  - subject( )
  - issuer( )
  - valid_not_before( )
  - valid_not_after( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::CACertificate - IF-MAP CA Certificate object.

DESCRIPTION

You can configure the Infoblox appliance as a metadata access point (MAP) and implement the IF-MAP protocol defined by the Trusted Computing Group (TCG, www.trustedcomputinggroup.com). This module provides an interface for configuring CA certificates that the IF-MAP server uses to authenticate IF-MAP clients.

CONSTRUCTOR

```
my $ca_cert=Infoblox::IFMap::CACertificate->new(
    certificate      => $string, #Required
);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to an IF-MAP CA Certificate object.

Infoblox::Session->add( )

Use this method to add a new CA certificate for the IF-MAP server. See Infoblox::Session->add( ) for parameters and return values.

Example

```
# Read the certificate contents from the file
open CERTFILE "<cert.pem" or die "Unable to open certificate file";
my $pem_cert = join('', <CERTFILE>);

# Create a new CA certificate object
my $cacert = Infoblox::IFMap::CACertificate->new(
certificate => $pem_cert
);

# Add the CA certificate
my $response = $session->add( $cacert );
```

Infoblox::Session->get( )

Use this method to retrieve the CA certificates configured for the IF-MAP server. See Infoblox::Session->get( ) for parameters and return values.

Key References

Apply the following attributes to get a specific IF-MAP CA certificate object:

```
subject                 - Optional. String that contains a certificate subject.
issuer                  - Optional. String that contains a certificate issuer.
```

```
 valid_not_after        - Optional. String that contains a desired value of the certificate
expiration date in ISO 8601 format.
 valid_not_before       - Optional. String that contains a desired value of the certificate start
date in ISO 8601 format.
```

Example

```
# Retrieve CA certificate with the subject = "/CN=verisign.com/OU=Verisign"
my @retrieved_objs = $session->get(
object => "Infoblox::IFMap::CACertificate",
subject => "/CN=verisign.com/OU=Verisign"
);
```

Infoblox::Session->remove( )

Use this method to remove a CA certificate.

Examples

```
# Get the CA certificate
my @retrieved_objs = $session->get(
object => "Infoblox::IFMap::CACertificate",
subject   => "/CN=verisign.com/OU=Verisign"
);

# Find the desired object from the retrieved list.
my $desired_cert = $retrieved_objs[0];

# Submit for removal
my $response = $session->remove( $desired_cert );
```

Infoblox::Session->search( )

Use this method to search for the CA certificates configured for the IF-MAP server. You can specify the following attributes in a search: subject, issuer, valid_not_before, and valid_not_after.

Key References

Apply the following attributes to search for a specific IF-MAP CA certificate object:

```
 subject                - Optional. Regex for the certificate subject.
 issuer                 - Optional. Regex for the certificate issuer.
 valid_not_after        - Optional. String that contains a desired value of the certificate
expiration date in ISO 8601 format.
 valid_not_before       - Optional. String that contains a desired value of the certificate start
date in ISO 8601 format.
 valid_not_after_start  - Optional. Interval search for valid_not_after attribute. Specify start of
the interval in ISO 8601 format.
 valid_not_after_end    - Optional. Interval search for valid_not_after attribute. Specify end of the
interval in ISO 8601 format.
 valid_not_before       - Optional. String that contains a desired value of the certificate start
time in ISO 8601 format (exact search).
 valid_not_before_start - Optional. Interval search for valid_not_before attribute. Specify start of
the interval in ISO 8601 format.
 valid_not_before_end   - Optional. Interval search for valid_not_before attribute. Specify end of
the interval in ISO 8601 format.
```

Examples

```
# Search for IF-MAP CA certificates issued by Verisign
# and with valid not before time in 2010
my @retrieved_objs = $session->search(
object => "Infoblox::IFMap::CACertificate",
issuer => ".*verisign.*",
valid_not_before_start => "2010-01-01T00:00:00Z",
valid_not_before_end => "2011-01-01T00:00:00Z",
);
```

METHODS

certificate( )

Use this method to retrieve or set the value of the certificate attribute. If no argument is passed, the method returns the current attribute value. The value must be a single certificate in PEM format.

Parameter

The valid value is a string that contains the certificate in PEM format. This is a required attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get certificate
my $pem_cert = $cert_object->certificate( );

# Set certificate
open CERTFILE "<cert.pem" or die "Unable to open certificate file";
my $pem_cert = join('', <CERTFILE>);
$cert_object->certificate( $pem_cert );
```

subject( )

Use this method to retrieve the value of the subject attribute. This method does not support any argument because the attribute is read-only and is set by the server. It has a value only when it reads an existing certificate from the Infoblox::IFMap::CACertificate object retrieved from the appliance.

Parameter

N/A.

Returns

The method returns the attribute value.

This is a read-only attribute. If you specify a parameter, the method returns an error.

Examples

```
# Get subject
my $subject = $certificate->subject( );
```

issuer( )

Use this method to retrieve the value of the =issuer= attribute. This method does not support any argument because the attribute is read-only and is set by the server. It has a value only when it reads an existing certificate from the =Infoblox::IFMap::CACertificate= object.

Parameter

N/A.

Returns

The method returns the attribute value.

This is a read-only attribute. If you specify a parameter, the method returns an error.

Examples

```
# Get subject
my $issuer = $certificate->issuer( );
```

valid_not_before( )

Use this method to retrieve the value of the valid_not_before attribute. This method does not support any argument because the attribute is read-only and is set by the server. It has a value only when it reads an existing certificate from the Infoblox::IFMap::CACertificate object. The returned value is the date before the certificate becomes invalid in ISO 8601 format.

Parameter

N/A.

Returns

The method returns the attribute value.

This is a read-only attribute. If you specify a parameter, the method returns an error.

Examples

```
# Get valid_not_before attribute
my $valid_not_before = $certificate->valid_not_before( );
```

valid_not_after( )

Use this method to retrieve the value of the valid_not_after attribute. This method does not support any argument because it is read-only and is set by the server. It has a value only when it reads an existing certificate from the Infoblox::IFMap::CACertificate object. The returned value is the date after the certificate becomes invalid in the ISO 8601 format.

Parameter

N/A.

Returns

The method returns the attribute value.

This is a read-only attribute. If you specify a parameter, the method returns an error.

Examples

```
# Get operation
my $valid_not_after = $certificate->valid_not_after( );
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search and remove. This sample also includes error handling for the operations.

#Preparations

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;
#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
);
unless ($session) {
       die("Construct session failed: ",
            Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "Session created successfully\n";

#Get a certificate from the external file
local *CERTFILE;
open CERTFILE, "<API_unit_test_sample_code/ca_cert.pem" or die "Unable to open certificate file";
my $pem_cert = join('', <CERTFILE>);
close(CERTFILE);
```

#Create CA certificate object

```
my $ca_certificate= Infoblox::IFMap::CACertificate->new(
                                                         certificate => $pem_cert
                                                       );
unless($ca_certificate) {
       die("Construct CA certificate object failed: ",
               Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "IFMap CA Certificate object created successfully\n";
```

#Add the CA Certificate object to the Infoblox appliance through a session

```
$session->add($ca_certificate)
    or die("Add IFMap CA certificate object failed: ",
                        $session->status_code( ) . ":" . $session->status_detail( ));
print "IFMap CA Certificate object added to server successfully\n";

#Get the IFMap CA Certificate object using subject attribute
my @retrieved_objs=$session->get(
                                 object => "Infoblox::IFMap::CACertificate",
                                 subject => 'CN="Test CA 2",OU="Engineering",O="Infoblox",L="Santa
Clara",ST="CA",C="US"'
                               );
my $object = $retrieved_objs[0];
unless($object){
    die("Get IFMap CA Certificate object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
}
```

```
print "Get IFMap CA Certificate object found at least 1 matching entry\n";
```

#Search for IFMap CA certificate objects

```
my @retrieved_objs=$session->search(
                                  object => "Infoblox::IFMap::CACertificate",
                                  subject => '.*Engineering.*',
                                  valid_not_before_start => '2010-01-01T08:00:00Z',
                                  valid_not_before_end => '2010-10-01T00:00:00Z'
                                );
my $object = $retrieved_objs[0];
unless($object){
    die("Search for IFMap CA Certificate object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Searching IFMap CA Certificate object found at least 1 matching entry\n";
```

#Remove IFMap CA Certificate object

```
#Get IFMap CA Certificate object
my @retrieved_objs=$session->get(
                                  object => "Infoblox::IFMap::CACertificate",
                                  subject => 'CN="Test CA 2",OU="Engineering",O="Infoblox",L="Santa
Clara",ST="CA",C="US"'
                                );
my $object = $retrieved_objs[0];
unless($object){
    die("Get IFMap CA certificate object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Get IFMap CA certificate object found at least 1 matching entry\n";

#submit the object for removal
$session->remove($object)
    or die("Remove IFMap CA certificate failed: ",
            $session->status_code( ) . ":" . $session->status_detail( ));
print "IFMap CA certificate object removed successfully \n";
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add( ), Infoblox::Session->get( ), Infoblox::Session->remove( ), Infoblox::Session->search( )

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::IFMap::Client - IF-MAP Client object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - authentication( )
  - authorization_rules( )
  - certificate( )
  - comment( )
  - disabled( )
  - extensible_attributes( )
  - name( )
  - override_session_timeout( )
  - override_subscription_buffer_size( )
  - password( )
  - session_remaining_time( )
  - session_timeout( )
  - status( )
  - subscription_buffer_size( )
  - subscription_remaining_buffer_size( )
  - username( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Client - IF-MAP Client object

DESCRIPTION

The Infoblox appliance can function as a metadata access point (MAP) and implement the IF-MAP protocol defined by the Trusted Computing Group (TCG, www.trustedcomputinggroup.com). This module provides an interface for creating and configuring credentials and access rights for IF-MAP clients.

CONSTRUCTOR

```
 my $ifmapclient = Infoblox::IFMap::Client->new(
     authentication                   => "basic" | "certificate",        #Required
     certificate                      => $cert,                          #Required if authentication =
"certificate"
     name                             => $string,                        #Required
     password                         => $string,                        #Required if authentication =
"basic"
     username                         => $string,                        #Required if authentication =
"basic"
     authorization_rules              => [ $rule1, $rule2, ...],         #Optional. Default is empty list
     comment                          => $string,                        #Optional
     disabled                         => "true" | "false",               #Optional / Default is "false"
     session_timeout                  => $integer,                       #Optional / Default taken from
Infoblox::IFMap::Grid
     override_session_timeout         => "true" | "false",               #Optional / Default is "false"
     subscription_buffer_size         => $integer,                       #Optional / Default taken from
Infoblox::IFMap::Grid
     override_subscription_buffer_size => "true" | "false",              #Optional / Default is "false"
     extensible_attributes => { $attrdef1 => $attrval1, $attrdef2 => $attrval2, ...}, #Optional
   );
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to an IF-MAP Client object.

Infoblox::Session->add( )

Use this method to add a new IF-MAP client. See Infoblox::Session->add( ) for parameters and return values.

Example

```
# Create a new IF-MAP Client object
my $ifmapclient = Infoblox::IFMap::Client->new(
     name => "client1",
     authentication => "basic",
     password => "secret"
     username => "user1"
     );

# Add the client
my $response = $session->add( $ifmapclient );
```

Infoblox::Session->get( )

Use this method to retrieve the IF-MAP clients. See Infoblox::Session->get( ) for parameters and return values.

Key References

```
Apply the following attributes to get a specific IF-MAP Client object:

name                   - Optional. String containing the name of the object.
comment                - Optional. String containing a comment.
extensible_attributes  - Optional. A hash reference containing extensible attributes.
```

Examples

```
# Retrieve the IF-MAP client with the name = "client1"
my @retrieved_objs = $session->get(
     object => "Infoblox::IFMap::Client",
     name => "client1"
     );
```

Infoblox::Session->modify( )

Use this method to update an IF-MAP client. This methods commits the changes applied to the Infoblox::IFMap::Client object. See Infoblox::Session->modify( ) for parameters and return values.

Examples

```
# Disable IF-MAP client
$ifmapclient->disabled( "true" );

# Update IF-MAP client
my $response = $session->modify( $ifmapclient );
```

Infoblox::Session->remove( )

Use this method to remove an IF-MAP client.

Examples

```
# Get the IF-MAP client
my @retrieved_objs = $session->get(
     object => "Infoblox::IFMap::Client",
     name   => "client1"
     );

# Find the desired object from the retrieved list.
my $desired_client = $retrieved_objs[0];

# Submit for removal
my $response = $session->remove( $desired_client );
```

Infoblox::Session->search( )

Use this method to search for IF-MAP clients. The following attributes can be specified for the search: name, comment, extensible_attributes.

Key References

```
Apply the following attributes to get a specific IF-MAP Client object:

name                   - Optional. Regex for the name of the object.
```

```
comment                - Optional. Regex for comment.
extensible_attributes  - Optional. A hash reference containing extensible attributes.
```

Examples

```
# Search for IF-MAP clients with names beginning with "cl"
my @retrieved_objs = $session->search(
     object => "Infoblox::IFMap::Client",
     name => "cl.*"
     );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an IF-MAP client.

authentication( )

Use this method to set or retrieve the value of the authentication attribute. This determines if a client authenticates using its own certificate or basic authentication credentials. Depending on the value of this attribute, either the password and username must be set or a certificate must be uploaded.

Parameter

Available values for the parameter:

```
"basic"       -  Username and password authentication
"certificate" -  Certificate authentication
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get authentication
my $authentication = $ifmapclient->authentication( );

# Set authentication
$ifmapclient->authentication( "certificate" );
```

authorization_rules( )

Use this method to set or retrieve the value of the authorization_rules attribute. This determines which action this client is allowed to perform by the IF-MAP server. This is an ordered list with the "first match" rule and the default is deny. If authorization rules are defined at the system level, they may apply to this client as well.

Parameter

Valid value is an array reference that contains Infoblox::IFMap::AuthRule objects. The default value is an empty list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get authorization_rules
my $ref_authorization_rules = $ifmapclient->authorization_rules( );

# Create an authorization rule preventing clients from publishing anything
my $rule = Infoblox::IFMap::AuthRule->new(
     operation  => "publish",
     identifier => "all",
     permission => "deny"
     );

# Set authorization_rules
$ifmapclient->authorization_rules( [$rule] );
```

certificate( )

Use this method to set or retrieve the value of the certificate attribute. This certificate is used to authenticate the IF-MAP client when it connects to the IF-MAP server.

Parameter

The valid value contains the Infoblox::IFMap::Client::Certificate object. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get certificate
my $certificate = $ifmapclient->certificate( );

# Create certificate
open CERTFILE "<cert.pem" or die "Unable to open certificate file";
my $pem_cert = join('', <CERTFILE>);

my $newcert = Infoblox::IFMap::Client::Certificate->new(
    certificate => $pem_cert
    );

# Set certificate
$ifmapclient->certificates( $newcert );
```

comment( )

Use this method to set or retrieve the value of the comment attribute.

Parameter

Desired comment in string format with a maximum of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get comment
my $comment = $ifmapclient->comment( );

# Set comment
$ifmapclient->comment( "new comment" );
```

disabled( )

Use this method to set or retrieve the value of the disabled attribute. If the disabled flag is set to "true", the IF-MAP server won't allow this client to connect.

Parameter

Specify "true" to set the disabled flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get disabled
my $disabled = $ifmapclient->disabled( );

# Disable IF-MAP client
$ifmapclient->disabled( "true" );
```

extensible_attributes( )

Use this method to set or retrieve extensible attributes for this object. If no argument is passed, it returns the list of current attributes that are set.

Parameter

The valid value is a hash reference containing the names of extensible attributes and their associated values. The names must

be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get extensible attributes
my $ref_extensible_attributes = $ifmapclient->extensible_attributes( );

# Modify extensible attributes
$ifmapclient->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom'
] });
```

name( )

Use this method to set or retrieve the value of the name attribute. This name uniquely identifies a client.

Parameter

Client name. A name can have a maximum of 255 characters.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get name
my $name = $ifmapclient->name( );

# Set name
$ifmapclient->name( "newname" );
```

override_session_timeout( )

Use this method to retrieve or set the value of the override_session_timeout attribute. If no argument is passed, the method returns the current value. This indicates whether the session_timeout value is overridden at the client level or inherited from the system properties.

Parameter

Override session timeout in boolean format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get override session timeout
my $override = $ifmapclient->override_session_timeout( );

# Set override session timeout
$ifmapclient->override_session_timeout( "true" );
```

override_subscription_buffer_size( )

Use this method to retrieve or set the value of the override_subscription_buffer_size attribute. If no argument is passed, it returns the current value. This indicates whether the subscription_buffer_size value is overridden at the client level or inherited from the system properties.

Parameter

Override subscription buffer size in boolean format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get override subscription buffer size
my $override = $ifmapclient->override_subscription_buffer_size( );

# Set override subscription buffer size
$ifmapclient->subscription_buffer_size( "true" );
```

password( )

Use this method to set the value of the password attribute. This value is not returned by the server when the get( ) and search( ) session methods are used. This password is used for basic authentication and is required if authentication is set to "basic".

Parameter

Password in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value. Note that it is possible to read passwords for newly created objects only. It is not possible to retrieve passwords from the database.

Examples

```
# Set password
$ifmapclient->password( "secret" );
```

session_remaining_time( )

Use this method to check the remaining idle time before a client session times out. This method does not support any argument because the attribute is read-only and is set by the server. It has a value only when it reads an existing Infoblox::IFMap::Client object and when Infoblox::IFMap::Client->`status()` returns DISCONNECTED. In other cases, it returns 0.

Parameter

Session remaining time in seconds in integer format.

Returns

This method returns the attribute value.

Examples

```
# Get session remaining time
my $ttl = $ifmapclient->session_remaining_time( );
```

session_timeout( )

Use this method to retrieve or set the IF-MAP client session timeout value at the client level. If no argument is passed, the method returns the current attribute value.

Parameter

Session timeout in seconds in integer format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get session timeout
my $timeout = $ifmapclient->session_timeout( );

# Set session timeout
$ifmapclient->session_timeout( 300 );
```

status( )

Use this method to check if an IF-MAP client is connected to the IF-MAP server. This method does not support any argument because the attribute is read-only and is set by the server. It has a value only when it reads an existing Infoblox::IFMap::Client object. The returned value can be CONNECTED or DISCONNECTED.

Parameter

Session connection status in boolean format.

Returns

This method returns the attribute value.

Examples

```
# Get connection status
my $status = $ifmapclient->status( );
```

subscription_buffer_size( )

Use this method to retrieve or set the client connection buffer size at the client level. The IF-MAP server stores subscription updates in a buffer for each client session. If no argument is passed, the method returns the current attribute value.

Parameter

Subscription buffer size in KB in integer format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get subscription buffer size
my $buffer_size = $ifmapclient->subscription_buffer_size( );

# Set subscription buffer size
$ifmapclient->subscription_buffer_size( 250 );
```

subscription_remaining_buffer_size( )

Use this method to check the remaining memory in the buffer for storing subscription updates before the server empites the buffer. This method does not support any argument because the attribute is read-only and is set by the server. It has a value only when it reads an existing Infoblox::IFMap::Client object.

Parameter

Subscription remaining buffer size in KB in integer format.

Returns

This method returns the attribute value.

Examples

```
# Get subscription remaining buffer size
my $remaining_buff_size = $ifmapclient->subscription_remaining_buffer_size( );
```

username( )

Use this method to set or retrieve the value of the username attribute. If no argument is passed, it returns the current value. This username is used for basic authentication and is required if authentication is set to "basic". It must be unique.

Parameter

Username in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get username
my $username = $ifmapclient->username( );

# Set username
$ifmapclient->username( "user1" );
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparations

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
               master   => "192.168.1.2", #appliance host ip
               username => "admin",       #appliance user login
               password => "infoblox"     #appliance password
               );

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "Session created successfully\n";
```

#Create an IF-MAP Client object with basic authentication

```
my $client_basic = Infoblox::IFMap::Client->new(
                       name     => "testclient_basic",
                       comment  => "test client 1",
                       authentication => "basic",
                       username => "user1",
                       password => "_secure!",
                     );

unless ($client_basic) {
       die("Construct client object failed: ",
               Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "IF-MAP Client object created successfully\n";

#Add the IF-MAP Client object to the Infoblox appliance through a session
$session->add($client_basic)
       or die("Add IF-MAP Client object failed: ",
                       $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object added to server successfully\n";
```

#Search for an IF-MAP Client

```
my @retrieved_objs = $session->search(
               object => "Infoblox::IFMap::Client",
               name   => "test.*"
               );
my $object = $retrieved_objs[0];

unless ($object) {
       die("Search IF-MAP Client object failed: ",
               $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Search IF-MAP Client object found at least 1 matching entry\n";
```

#Get and modify an IF-MAP Client object

```
#Get the IF-MAP Client object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Client",
    name   => "testclient_basic"
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get IF-MAP Client object failed: ",
               $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Get IF-MAP Client object found at least 1 matching entry\n";
```

```
#Modify the IF-MAP Client object
$object->name("testclient_renamed");
$object->comment("this is a new comment");

#Apply the changes.
$session->modify($object)
       or die("Modify IF-MAP Client object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object modified successfully \n";

#Accessor methods
print "name: " . $object->name( ) . "\n";
print "comment: " . $object->comment( ) . "\n";
```

#Remove an IF-MAP Client object

```
#Get the IF-MAP Client object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Client",
    name   => "testclient_renamed"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get IF-MAP Client object failed: ",
        $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Get IF-MAP Client object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove IF-MAP Client object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object removed successfully \n";
```

#Get and modify an IF-MAP Client object's session timeout

```
#Get the IF-MAP Client object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Client",
    name   => "testclient_session_to"
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get IF-MAP Client object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Get IF-MAP Client object found at least 1 matching entry\n";

#Modify the IF-MAP Client object's session timeout
$object->override_session_timeout("true");
$object->session_timeout(300);

#Apply the changes.
$session->modify($object)
       or die("Modify IF-MAP Client object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object modified successfully \n";

#Accessor methods
print "override session timeout: " . $object->override_session_timeout( ) . "\n";
print "session timeout: " . $object->session_timeout( ) . "\n";
```

#Get and modify an IF-MAP Client object's subscription buffer size

```
#Get the IF-MAP Client object from Infoblox appliance through a session
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Client",
    name   => "testclient_session_to"
);
my $object = $retrieved_objs[0];

unless ($object) {
       die("Get IF-MAP Client object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Get IF-MAP Client object found at least 1 matching entry\n";
```

```
#Modify the IF-MAP Client object's subscription buffer size
$object->override_subscription_buffer_size("true");
$object->subscription_buffer_size(300);

#Apply the changes.
$session->modify($object)
       or die("Modify IF-MAP Client object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object modified successfully \n";

#Accessor methods
print "override subscription buffer size: " . $object->override_subscription_buffer_size( ) . "\n";
print "subscription buffer size: " . $object->subscription_buffer_size( ) . "\n";
```

#Create an IF-MAP Client object with certificate authentication

```
# Create certificate
local *CERTFILE;
open CERTFILE, "<API_unit_test_sample_code/cert.pem" or die "Unable to open certificate file";
my $pem_cert = join('', <CERTFILE>);
close(CERTFILE);
my $ifmap_cert = Infoblox::IFMap::Client::Certificate->new(
                    certificate => $pem_cert
                  );
unless($ifmap_cert) {
       die("Construct certificate object failed: ",
                Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "IF-MAP Client Certificate object created successfully\n";

my $client_cert = Infoblox::IFMap::Client->new(
                      name     => "testclient_certified",
                      comment  => "test client 2",
                      authentication => "certificate",
                      certificate => $ifmap_cert
                    );

unless ($client_cert) {
       die("Construct client object failed: ",
                Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "IF-MAP Client object created successfully\n";

#Add the IF-MAP Client object to the Infoblox appliance through a session
$session->add($client_cert)
       or die("Add IF-MAP Client object failed: ",
                      $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object added to server successfully\n";
```

#Remove an IF-MAP Client object

```
#Get the IF-MAP Client object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Client",
    name   => "testclient_certified"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get IF-MAP Client object failed: ",
        $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Get IF-MAP Client object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove IF-MAP Client object failed: ",
                $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object removed successfully \n";
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::AuthRule, Infoblox::IFMap::Client::Certificate, Infoblox::Session, Infoblox::Session->add( ), Infoblox::Session->get( ), Infoblox::Session->modify( ), Infoblox::Session->remove( ), Infoblox::Session->search( )

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::IFMap::Client::Certificate - IF-Map Client Certificate object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::IFMap::Client->certificate( )
- METHODS
  - certificate( )
  - subject( )
  - issuer( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Client::Certificate - IF-Map Client Certificate object

DESCRIPTION

This module provides an interface for configuring certificates that are used to authenticate IF-MAP clients.

CONSTRUCTOR

```
my $cert_object = Infoblox::IFMap::Client::Certificate->new(
                  certificate => $string,                    #Required
);
```

MODULE METHODS

Infoblox::IFMap::Client->certificate( )

Use this method to set or retrieve the certificate of a given IF-MAP client. (See Infoblox::IFMap::Client)

Examples

```
# Get certificate
my $ref_certificates = $ifmapclient->certificates( );

# Set certificate
$ifmapclient->certificates( $newcert );
```

METHODS

certificate( )

Use this method to set or retrieve the value of the certificate attribute. The value must be a single certificate in PEM format.

Parameter

String containing the certificate. The certificate should be in PEM format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
# Get certificate
my $pem_cert = $cert_object->certificate( );

# Set certificate
open CERTFILE "<cert.pem" or die "Unable to open certificate file";
my $pem_cert = join('', <CERTFILE>);
$cert_object->certificate( $pem_cert );
```

subject( )

Use this method to retrieve the value of the subject attribute. This method does not support any argument as this attribute is read-only and is set by the server. It has a value only when reading an existing certificate from an Infoblox::IFMap::Client object.

Parameter

This is a read-only attribute.

Returns

When called without parameters, the method returns the subject attribute of the certificate.

Calling this method with parameters results in an error.

Examples

```
# Get subject
my $subject = $certificate->subject( );
```

issuer( )

Use this method to retrieve the value of the issuer attribute. This method does not support any argument because the attribute is read-only and is set by the server. It has a value only when it reads an existing certificate from the Infoblox::IFMap::Client object.

Parameter

This is a read-only attribute.

Returns

When you do not specify a parameter, the method returns the issuer attribute of the certificate.

When you specify a parameter, the method returns an error.

Examples

```
# Get issuer
my $issuer = $certificate->issuer( );
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to the object. This sample also includes error handling for the operations.

#Preparations

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
              master   => "192.168.1.2", #appliance host ip
              username => "admin",       #appliance user login
              password => "infoblox"     #appliance password
              );

unless ($session) {
       die("Construct session failed: ",
              Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "Session created successfully\n";
```

#Create an IF-MAP Client object with certificate authentication

```
# Create certificate
local *CERTFILE;
open CERTFILE, "<./cert.pem" or die "Unable to open certificate file";
my $pem_cert = join('', <CERTFILE>);
close(CERTFILE);
my $ifmap_cert = Infoblox::IFMap::Client::Certificate->new(
                     certificate => $pem_cert
                 );
```

```
unless($ifmap_cert) {
       die("Construct certificate object failed: ",
               Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "IF-MAP Client Certificate object created successfully\n";

my $client_cert = Infoblox::IFMap::Client->new(
                       name     => "testclient_certified",
                       comment  => "test client 2",
                       authentication => "certificate",
                       certificate => $ifmap_cert
                   );

unless ($client_cert) {
       die("Construct client object failed: ",
               Infoblox::status_code( ) . ":" . Infoblox::status_detail( ));
}
print "IF-MAP Client object created successfully\n";

#Add the IF-MAP Client object to the Infoblox appliance through a session
$session->add($client_cert)
       or die("Add IF-MAP Client object failed: ",
                       $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object added to server successfully\n";
```

#Remove an IF-MAP Client object

```
#Get the IF-MAP Client object through the session
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Client",
    name   => "testclient_certified"
);
my $object = $retrieved_objs[0];
unless ($object) {
    die("Get IF-MAP Client object failed: ",
        $session->status_code( ) . ":" . $session->status_detail( ));
}
print "Get IF-MAP Client object found at least 1 matching entry\n";

#Submit the object for removal
$session->remove($object)
       or die("Remove IF-MAP Client object failed: ",
               $session->status_code( ) . ":" . $session->status_detail( ));
print "IF-MAP Client object removed successfully \n";
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::Client

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::IFMap::Client::Credentials - IF-MAP Credentials object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::IFMap::Federation::Connector->credentials( )
- METHODS
  - authentication( )
  - password( )
  - server_ca_cert( )
  - server_url( )
  - username( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Client::Credentials - IF-MAP Credentials object

DESCRIPTION

This object represents the credentials used by an IF-MAP Federation Connector object.

CONSTRUCTOR

```
my $credentials = Infoblox::IFMap::Federation::Connector::Credentials->new(
    authentication => <"basic" | "certificate">,                #Required
    password => <STRING>,                                       #Required if authentication = "basic"
    server_ca_cert => $cacert,                                  #Required
    server_url => <STRING>,                                     #Required
    username => <STRING>,                                       #Required if authentication = "basic"
);
```

MODULE METHODS

The following functions are available to apply to an IF-MAP Credentials object.

Infoblox::IFMap::Federation::Connector->credentials( )

Use this function to set or retrieve the credentials for a given connector. See Infoblox::IFMap::Federation::Connector for parameters and return values.

Example

```
# Note that this certificate must have already been added to the server
my $cert_obj=Infoblox::IFMap::CACertificate->new(
                                                 certificate => $cert_pem,
                                                );

# Credentials
my $credentials = Infoblox::IFMap::Client::Credentials->new(
        authentication => "basic",
        password => "secret",
        username => "user1",
        server_ca_cert => $cert_obj,
        server_url => "https://somewhere.com/",
        );

# Set attributes
$connector->credentials($credentials);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of an IF-MAP Credentials object.

authentication( )

Use this method to set or retrieve the value of the authentication attribute.

Parameter

A string that represents the attribute. Valid strings are "basic" or "certificate".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the authentication attribute.

Example

```
# Get authentication
my $authentication = $credentials->authentication( );

# Change it
$credentials->authentication("basic");
```

password( )

Use this method to set the value of the password attribute. This value is not returned by the server when you use the `get()` and `search()` session methods. This password is used for Basic Authentication and is required if authentication is set to "basic".

Parameter

A string that specifies the password.

Example

```
# Set it
$credentials->password("secret");
```

server_ca_cert( )

Use this method to set or retrieve the value of the server_ca_cert attribute. This configures the CA certificate to use for server certificate validation.

Parameter

An Infoblox::IFMap::CACertificate object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute server_ca_cert.

Example

```
# Get server_ca_cert
my $server_ca_cert = $credentials->server_ca_cert( );

# Change it
$credentials->server_ca_cert($ca_cert);
```

server_url( )

Use this method to set or retrieve the value of the server_url attribute. It specifies the URL of the federated IF-MAP service.

Parameter

The URL of the IF-MAP service. For example "https://ifmap.infoblox.com:443/ifmap".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute server_url.

Example

```
# Get server_url
my $server_url = $credentials->server_url( );

# Change it
$credentials->server_url("https://ifmap.infoblox.com:443/ifmap");
```

username( )

Use this method to set or retrieve the value of the username attribute. This username is used for Basic Authentication and is required if authentication is set to "basic". It has to be unique per server, specified by the server_url attribute.

Parameter

A string that represents the attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute username.

Example

```
# Get username
my $username = $credentials->username( );

# Change it
$credentials->username("user1");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::Federation::Connector, Infoblox::IFMap::CACertificate

---

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::IFMap::Federation::Connector - IF-MAP Connector object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - credentials( )
  - disabled( )
  - extensible_attributes( )
  - name( )
  - override_retry_interval( )
  - publisher_id( )
  - purge_on_startup( )
  - retry_interval( )
  - status( )
  - status_detail( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Federation::Connector - IF-MAP Connector object

DESCRIPTION

This object represents an IF-MAP federation connector.

CONSTRUCTOR

```
my $fedcon = Infoblox::IFMap::Federation::Connector->new(
    credentials => $credentials,                                  #Required
    name => $string,                                              #Required
    comment => $string,                                           #Optional / Default is empty
    disabled => <"true" | "false">,                               #Optional / Default is false
    extensible_attributes => { $attrdef1 => $attrval1, ...},      #Optional / Default is empty,
    purge_on_startup => <"true" | "false">,                       #Optional / Default is true
    override_retry_interval => <"true" | "false">                 #Optional / Default is false
    retry_interval => $num,                                       #Optional
);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to an IF-MAP Federation Connector object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
# Create a new Federation Connector object
my $fedcon = Infoblox::IFMap::Federation::Connector->new(
    name => "fedcon1",
    credentials => $credentials,
    );

# Add the connector
my $response = $session->add( $fedcon );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific Federation Connector object:

 authentication - Optional. The valid value is either "basic" or "certificate".
 comment - Optional. A comment in string format.
 disabled - Optional. The valid value is either "true" or "false".
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 name - Optional. A name in string format.
 server_url - Optional. A server_url in string format.
 username - Optional. A username in string format.
```

Example

```
# Retrieve the federation connector with the  name "fedcon1"
my @retrieved_objs = $session->get(
         object => "Infoblox::IFMap::Federation::Connector",
         name => "fedcon1"
         );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Disable the federation connector
$fedcon->disabled( "true" );

# Update the federation connector
my $response = $session->modify( $fedcon );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the object, and then submit it for removal.

Example

```
# Get the Federation Connector object
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Federation::Connector",
    name   => "fedcon1"
    );

# Find the desired object on the retrieved list.
my $desired_connector = $retrieved_objs[0];

# Submit for removal
my $response = $session->remove( $desired_connector );
```

Infoblox::Session->search( )

Use this method to search for Federation Connector objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a Federation Connector object

 authentication - Optional. Valid values are "basic" or "certificate".
 comment - Optional. A comment in string format (regular expression).
 disabled - Optional. Valid values are "true" or "false".
 extensible_attributes - Optional. A hash reference containing extensible attributes.
 name - Optional. A name in string format (regular expression).
 server_url - Optional. A server_url in string format (regular expression).
 username - Optional. A username in string format (regular expression).
```

Example

```
# Search for federation connectors with names that begin with "fedcon"
my @retrieved_objs = $session->search(
         object => "Infoblox::IFMap::Federation::Connector",
         name => "fedcon.*"
         );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an IF-MAP Federation Connector object.

comment( )

Use this method to set or retrieve the value of the comment attribute.

Parameter

A string with a maximum length of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $fedcon->comment();

# Set comment
$fedcon->comment( "Connector to retrieve authenticated users" );
```

credentials( )

Use this method to set or retrieve the value of the credentials attribute.

Parameter

An Infoblox::IFMap::Client::Credentials object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get credentials
my $credentials = $fedcon->credentials();

# Note that this certificate must have already been added to the server
my $cert_obj=Infoblox::IFMap::CACertificate->new(
                                                 certificate => $cert_pem,
                                                );

# Credentials
my $credentials = Infoblox::IFMap::Client::Credentials->new(
         authentication => "basic",
         password => "secret"
         username => "user1"
         server_ca_cert => $cert_obj,
         server_url => "https://somewhere.com/",
         );

# Set attributes
$fedcon->credentials($credentials);
```

disabled( )

Use this method to set or retrieve the value of the disabled attribute. If it is disabled, the federation connector won't poll its subscriptions.

Parameter

Specify "true" to set the disabled flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get disabled
my $disabled = $fedcon->disabled();

# Set disabled
$fedcon->disabled( "true" );
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with an IF-MAP Federation Connector object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference that contains the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $fedcon->extensible_attributes();
#Modify extensible attributes
$fedcon->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom' ] });
```

name( )

Use this method to set or retrieve the value of the name attribute. This name uniquely identifies a Federation Connector object.

Parameter

A string with a maximum length of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $name = $fedcon->name();

# Set name
$fedcon->name( "newname" );
```

override_retry_interval( )

The override_retry_interval attribute controls whether the retry_interval value in the Federation Connector object is used, instead of the system default.

The override_retry_interval attribute can be specified explicitly. It is also set implicitly when retry_interval is set to a defined value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the parameter to "true" to override the setting for retry_interval. Set the parameter to "false" to inherit the default setting for retry_interval.

The default value of this parameter is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Getting override_retry_interval
my $override_retry_interval=$fedcon->override_retry_interval( );
#Modifying override_retry_interval
$fedcon->override_retry_interval("true");
```

publisher_id( )

Use this method to retrieve the value of the publisher_id attribute. The publisher_id is the value assigned by the local IF-MAP server and allows searching for metadata published by this specific federation connector in the local database.

Parameter

none.

Returns

The method returns the attribute value.

Example

```
# Get publisher_id
my $publisher_id = $fedcon->publisher_id();
```

purge_on_startup( )

Use this method to set or retrieve the value of the purge_on_startup attribute. This indicates whether the connector purges data published in previous sessions or proceeds with updates without cleaning up older data.

Parameter

Specify "true" to set the purge_on_startup flag or "false" to deactivate/unset it. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get purge_on_startup
my $purge_on_startup = $fedcon->purge_on_startup();

# Set purge_on_startup
$fedcon->purge_on_startup( "false" );
```

retry_interval( )

Use this method to set or retrieve the value of the retry_interval attribute. This indicates the time, in seconds, between retries after a failed synchronization attempt. If the value is set to undef, federation connectors use the global setting defined in Infoblox::Grid::IFMap.

Setting this method to a defined value implicitly sets the override_retry_interval method to "true". Setting the parameter to undefined causes the appliance to use the global setting and automatically resets the override_retry_interval attribute to "false".

Note that when retry_interval is set to a defined value and override_retry_interval is set to "false", the last operation takes precedence. Thus the sequence $object->retry_interval(12); $object->override_retry_interval("false"); will set override_retry_interval to "false", and the sequence $object->override_retry_interval("false"); $object->retry_interval(12); will result in override_retry_interval="true".

Parameter

A number that represents the attribute. The value is expressed in seconds, and must not be less than five.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get retry_interval
my $retry_interval = $fedcon->retry_interval();

# Set retry_interval
$fedcon->retry_interval( 60 );
```

status( )

Use this method to retrieve the value of the status attribute. The returned value is "disabled", "running", "warning", or "failed".

Parameter

none

Returns

The method returns the attribute value.

Example

```
# Get status
my $status = $fedcon->status();
```

status_detail( )

Use this method to retrieve the value of the status_detail attribute.

Parameter

none

Returns

The method returns the attribute value.

Example

```
# Get status_detail
my $status_detail = $fedcon->status_detail();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, get, modify, search and remove. This sample also includes error handling for the operations.

#Preparation prior to getting and modifying an IF-MAP Federation Connector object

```
use strict;
use Infoblox;

#Refers to the Infoblox appliance IP address
my $host_ip = "192.168.1.2";

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
                                     master   => $host_ip,
                                     username => "admin",
                                     password => "infoblox"
                                    );
unless ($session) {
    die("Construct session failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";


B<#This needs to be modified to contain a valid certificate>
my $g_cert=<<EOA;
-----BEGIN CERTIFICATE-----



-----END CERTIFICATE-----
EOA

my $cert_obj=Infoblox::IFMap::CACertificate->new(
                                                 certificate => $g_cert
                                                );

$session->add($cert_obj) || die("Cert addition failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
```

```
B<#Create a new credential>
my $credential = Infoblox::IFMap::Client::Credentials->new(
                                                          authentication => 'certificate',
                                                          username       => 'hi',
                                                          password       => 'there',
                                                          server_url     => 'https://somewhere.com/',
                                                          server_ca_cert => $cert_obj,
                                                         );

B<#Create the connector object>
my $connector = Infoblox::IFMap::Federation::Connector->new(
                                                           comment                 => 'some comment',
                                                           credentials             => $credential,
                                                           disabled                => 'true',
                                                           extensible_attributes   => { Site =>
'Somewhere'},
                                                           name                    => 'connector name',
                                                          );

$session->add($connector) || die("Connector addition failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

B<#Search for the connector object>
my @retrieved_obj = $session->search(
  object => 'Infoblox::IFMap::Federation::Connector',
  name   => 'connector name',
);

my $t = $retrieved_obj[0];
$t->retry_interval(8);

B<#Modify the connector object>
$session->modify($t) || die("Connector modify failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

B<#Cleanup>
$session->remove($t) || die("Connector removal failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->remove($cert_obj) || die("Cert removal failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::IFMap::Client::Credentials

---

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::IFMap::Federation::Subscription - IF-MAP Subscription object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->add( )
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - comment( )
  - connector_name( )
  - disabled( )
  - extensible_attributes( )
  - match_links( )
  - max_depth( )
  - max_size( )
  - name( )
  - result_filter( )
  - start_point( )
  - terminal_identifier_types( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Federation::Subscription - IF-MAP Subscription object

DESCRIPTION

This object represents a subscription assigned to a Federation Connector object.

CONSTRUCTOR

```
my $subscription = Infoblox::IFMap::Federation::Subscription->new(
    connector_name => $string,           #Required
    name => $string,                     #Required
    start_point => $start_point,         #Required
    comment => $string,                  #Optional / Default is empty
    disabled => <"true" | "false">       #Optional / Default is false
    extensible_attributes => { $attrdef1 => $attrval1, ...},   #Optional / Default is empty
    match_links => $string,              #Optional / Default is undefined
    max_depth => $num,                   #Optional / Default is undefined
    max_size  => $num,                   #Optional / Default is undefined
    result_filter => $string,            #Optional / Default is undefined
    terminal_identifier_types => $string,#Optional / Default is empty
);
```

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to an IF-MAP Subscription object.

Infoblox::Session->add( )

Use this method to add an object to the Infoblox appliance. See Infoblox::Session->add() for parameters and return values.

Example

```
# Create an identifier
 my $identifier = Infoblox::IFMap::Identifier->new(
      type => "ip-address",
      name => "10.0.0.1"
      );

# Create a new Subscription object assigned to the Federation Connector object "fedcon1"
```

```
my $subs = Infoblox::IFMap::Federation::Subscription->new(
    name => "subs1",
    connector_name => "fedcon1",
    start_point => $identifier,
    result_filter => "meta:location",
    max_depth => 1
    );

# Add the subscription
my $response = $session->add( $subs );
```

Infoblox::Session->get( )

Use this method to retrieve all the matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key References

```
Apply the following attributes to get a specific subscription object:

 comment - Optional. A comment in string format (regular expression).
 connector_name - Required. A connector name in string format.
 name - Required. A subscription name in string format.
```

Example

```
# Retrieve a subscription named "subs1"  that is assigned to the Federation Connector object
"fedcon1"
my @retrieved_objs = $session->get(
         object => "Infoblox::IFMap::Federation::Subscription",
         name => "subs1",
         connector_name => "fedcon1"
         );
```

Infoblox::Session->modify( )

Use this method to modify an object in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
# Set maximum depth
$subs->max_depth( 2 );

# Update subscription
my $response = $session->modify( $subs );
```

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` or `search()` to retrieve the object, and then submit it for removal.

Example

```
# Get the subscription
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Federation::Connector",
    name => "subs1",
    connector_name   => "fedcon1"
    );

# Find the desired object on the retrieved list.
my $desired_subs = $retrieved_objs[0];

# Submit for removal
my $response = $session->remove( $desired_subs );
```

Infoblox::Session->search( )

Use this method to search for Federation Subscription objects in the Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key References

```
Apply the following attributes to search for a subscription object

 comment - Optional. A comment in string format (regular expression).
 connector_name - Required. A connector name in string format (regular expression).
 name - Required. A subscription name in string format (regular expression).
```

Example

```
# Search for subscriptions with names that begin with "sub" and that are assigned to the Federation Connector object "fedcon1"
my @retrieved_objs = $session->search(
        object => "Infoblox::IFMap::Federation::Subscription",
        connector_name => "fedcon1",
        name => "sub.*"
        );
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of an IF-MAP Subscription object.

comment( )

Use this method to set or retrieve the value of the comment attribute.

Parameter

A string with a maximum length of 256 bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get the comment
my $comment = $subscription->comment();

# Set a comment
$subscription->comment( "Subscription to retrieve authenticated users" );
```

connector_name( )

Use this method to set or retrieve the value of the connector_name attribute, which identifies the federation connector owner of this subscription.

Parameter

A string that represents the attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get connector name
my $name = $subscription->connector_name();

# Set connector name
$subscription->connector_name( "fedcon1" );
```

disabled( )

Use this method to set or retrieve the value of the disabled attribute. If the subscription is disabled, the federation connector won't poll this subscription.

Parameter

Specify "true" to set the disabled flag or "false" to deactivate/unset it. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get disabled
```

```
my $disabled = $subscription->disabled();

# Disable Federation Subscription
$subscription->disabled( "true" );
```

extensible_attributes( )

Use this method to set or retrieve the extensible attributes associated with an IF-MAP subscription.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is a hash reference that contains the names of extensible attributes and their associated values. The names must be those of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $subscription->extensible_attributes();
#Modify extensible attributes
$subscription->extensible_attributes({ 'Site' => 'Santa Clara', 'Administrator' => [ 'Peter', 'Tom'
] });
```

match_links( )

Use this method to set or retrieve the value of the match_links attribute.

Parameter

A string that defines an IF-MAP filter (subset of XPath), as specified in the IF-MAP protocol version 2.0. An empty value specifies a filter that does not match any metadata, and an undefined value (undef) specifies a filter that matches all metadata.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get match-links filter
my $filter = $subscription->match_links();

# Set match-links filter
$subscription->match_links( "meta:ip-mac or meta:authenticated-as" );
```

max_depth( )

Use this method to set or retrieve the value of the max_depth attribute. This limits the depth of the subscribed search.

Parameter

The valid value is any unsigned 32-bit integer that specifies the maximum depth for the search.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get maximum depth
my $depth = $subscription->max_depth();

# Set maximum depth
$subscription->max_depth( 3 );
```

max_size( )

Use this method to set or retrieve the value of the max_size attribute. This limits the result size of the subscribed search.

Parameter

The valid value is a number that specifies the maximum size for the search in bytes.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get maximum size
my $size = $subscription->max_size();

# Set maximum size (100 KB)
$subscription->max_size( 102400 );
```

name( )

Use this method to set or retrieve the value of the name attribute.

Parameter

A string that specifies the subscription name.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $name = $subscription->name();

# Set name
$subscription->name( "subs1" );
```

result_filter( )

Use this method to set or retrieve the value of the result_filter attribute.

Parameter

A string that defines an IF-MAP filter (subset of XPath), as specified in the IF-MAP protocol version 2.0. An empty value specifies a filter that does not match any metadata, and an undefined value (undef) specifies a filter that matches all metadata.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get result filter
my $filter = $subscription->result_filter();

# Set result filter
$subscription->result_filter( "meta:event or meta:unexpected-behavior" );
```

start_point( )

Use this method to set or retrieve the value of the start_point attribute, which defines the starting point of the IF-MAP search to which the federation connector subscribes.

Parameter

An Infoblox::IFMap::Identifier object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get start point
my $start_point = $subscription->start_point();

  # Create an identifier
  my $identifier = Infoblox::IFMap::Identifier->new(
       type => "ip-address",
       name => "10.0.0.1"
       );

# Set start point
$subscription->start_point( $identifier );
```

terminal_identifier_types( )

Use this method to set or retrieve the value of the terminal_identifier_types attribute, which tells the IF-MAP server to stop the search when any identifier of this type is reached.

Parameter

A string that represents the attribute, which is a list of comma-separated values. Valid values are ip-address, mac-address, device, identity, and access-request.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get terminal identifier type
my $terminal_types = $subscription->terminal_identifier_types();

# Set terminal identifier type
$subscription->terminal_identifier_types( "ip-address,device" );
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, get, modify, search and remove. This sample also includes error handling for the operations.

#Preparation prior to getting and modifying an IF-MAP Federation Subscription object

```
use strict;
use Infoblox;

#Refers to the Infoblox appliance IP address
my $host_ip = "192.168.1.2";

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
                                     #     master   => $host_ip,
                                     master   => '127.0.0.1',
                                     username => "admin",
                                     password => "infoblox"
                                    );
unless ($session) {
    die("Construct session failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

B<#This needs to be modified to contain a valid certificate>
my $g_cert=<<EOA;
-----BEGIN CERTIFICATE-----



-----END CERTIFICATE-----
EOA
```

```
my $cert_obj=Infoblox::IFMap::CACertificate->new(
                                                 certificate => $g_cert
                                                );

$session->add($cert_obj) || die("Cert addition failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

B<#Create a new identifier>
my $identifier = Infoblox::IFMap::Identifier->new(
                                                  administrative_domain     => 'some.domain.com',
                                                  ip_address_type           => 'IPv4',
                                                  name                      => '10.0.0.1',
                                                  type                      => 'ip-address',
                                                 );

B<#Create a new credential>
my $credential = Infoblox::IFMap::Client::Credentials->new(
                                                           authentication => 'certificate',
                                                           username       => 'hi',
                                                           password       => 'there',
                                                           server_url     => 'https://somewhere.com/',
                                                           server_ca_cert => $cert_obj,
                                                          );

B<#Create a new connector>
my $connector = Infoblox::IFMap::Federation::Connector->new(
                                                            comment                 => 'some comment',
                                                            credentials             => $credential,
                                                            disabled                => 'true',
                                                            name                    => 'connector name',
                                                           );

$session->add($connector) || die("Connector addition failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

B<#Create a new subscription>
my $subscription = Infoblox::IFMap::Federation::Subscription->new(
                                                                  comment                   => 'some
comment',
                                                                  connector_name            => 'connector
name',
                                                                  disabled                  => 'true',
                                                                  match_links               => 'meta:ip-mac
or meta:authenticated-as',
                                                                  max_depth                 => 3,
                                                                  max_size                  => 102400,
                                                                  name                      => 'subname',
                                                                  result_filter             => 'meta:event
or meta:unexpected-behavior',
                                                                  start_point               => $identifier,
                                                                  terminal_identifier_types => 'ip-
address,device',
                                                                  extensible_attributes     => { Site =>
'Somewhere'},
                                                                 );

$session->add($subscription) || die("Subscription addition failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

B<#Search for a subscription>
my @retrieved_obj = $session->search(
  object => 'Infoblox::IFMap::Federation::Subscription',
  name   => 'subname',
);

my $t = $retrieved_obj[0];
$t->match_links("meta:ip-mac");

B<#Modify the subscription>
$session->modify($t) || die("Subscription modify failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());


B<#Cleanup>
$session->remove($t) || die("Subscription removal failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->remove($connector) || die("Connector removal failed: ",
```

```
        Infoblox::status_code() . ":" . Infoblox::status_detail());
$session->remove($cert_obj) || die("Cert removal failed: ",
        Infoblox::status_code() . ":" . Infoblox::status_detail());

####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::Session->add(), Infoblox::Session->get(), Infoblox::Session->modify(), Infoblox::Session->remove(), Infoblox::Session->search(), Infoblox::IFMap::Identifier

---

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::IFMap::Identifier - IF-MAP Identifier object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::IFMap::Federation::Subscription->start_point( )
- METHODS
  - administrative_domain( )
  - custom_attributes( )
  - custom_identifier_type( )
  - device_type( )
  - identity_other_definition( )
  - identity_type( )
  - ip_address_type( )
  - name( )
  - type( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Identifier - IF-MAP Identifier object

DESCRIPTION

This object represents a standard IF-MAP identifier. It is used to specify the start point of a search while creating a subscription for federation connectors.

CONSTRUCTOR

```
 my $identifier = Infoblox::IFMap::Identifier->new(
     type => <"ip-address" | "mac-address" | "device" | "identity" | "access-request"
            | "custom">,                                                          #Required
     administrative_domain => $string,                                             #Optional /
Default is undef
     custom_attributes => [$attr1, $attr2, ...],                                   #Optional /
Default is undef
     custom_identifier_type => $cit,                                               #Optional /
Default is undef
     device_type => <"name" | "aik-name">,                                         #Optional /
Default is "name"
     identity_other_definition => $string,                                         #Optional /
Default is empty
     identity_type => <"aik-name" | "distinguished-name" | "dns-name" | "email-address"
                    | "kerberos-principal" | "username" | "sip-uri" | "tel-uri"
                    | "hip-hit" | "other">,                                       #Optional /
Default is "username"
     ip_address_type  => <"IPv4" | "IPv6">                                        #Optional /
Default is "IPv4"
     name => $string,                                                              #Optional / Default
is empty
 );
```

MODULE METHODS

The following functions can be applied to an IF-MAP Identifier object.

Infoblox::IFMap::Federation::Subscription->start_point( )

Use this method to retrieve or set the IF-MAP identifier start point of a subscribed search. See Infoblox::IFMap::Federation::Subscription for parameters and return values.

Example

```
       #Create the identifier
       my $identifier = Infoblox::IFMap::Identifier->new(
                   type => "ip-address",
                   name => "10.0.0.1"
```

```
            );

# Set it
$subscription->start_point($identifier);
```

METHODS

This section describes all the methods that you can use to set and retrieve the attribute values of an IF-MAP Identifier object.

administrative_domain( )

Use this method to set or retrieve the value of the administrative_domain attribute.

Parameter

A string that represents the attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get administrative_domain
my $administrative_domain = $identifier->administrative_domain( );

# Change it
$identifier->administrative_domain("infoblox");
```

custom_attributes( )

Use this method to set or retrieve the list of attributes for a custom identifier.

Parameter

A reference to an array of Infoblox::IFMap::Identifier::Attribute objects

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get attributes
my $attributes = $identifier->custom_attributes();

# Create an attribute
my $attr = Infoblox::IFMap::Identifier::Attribute->new(
                  name => "isbn",
                  value => "978-1565925120");
# Set attributes
$identifier->custom_attributes([$attr]);
```

custom_identifier_type( )

Use this method to set or retrieve the value of the custom_identifier_type attribute.

Parameter

An Infoblox::IFMap::Identifier::Definition object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get the custom identifier type
my $custom_type = $identifier->custom_identifier_type();

# Retrieve a custom identifier definition
my @objs = $session->get(
```

```
        object => "Infoblox::IFMap::Identifier::Definition",
        display_name => "Employee"
        );

my $iddef = $objs[0];

# Set the custom identifier type
$identifier->custom_identifier_type($iddef);
```

device_type( )

Use this method to set or retrieve the value of the device_type attribute. This attribute is meaningful only if the type attribute is set to "device".

Parameter

A string that represents the attribute. Valid values for this attribute are either "aik-name" or "name". The default is "name".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get device type
my $type = $identifier->device_type();

# Set device type
$identifier->device_type( "aik-name" );
```

identity_other_definition( )

Use this method to set or retrieve the value of the identity_other_definition attribute. This attribute is meaningful only if the type attribute is set to "identity" and the identity_type attribute is set to "other".

Parameter

A string that represents the attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get identity other type definition
my $definition = $identifier->identity_other_definition();

# Set identity other type definition
$identifier->identity_other_definition( "socialsecurity.gov:SSN" );
```

identity_type( )

Use this method to set or retrieve the value of the identity_type attribute. This attribute is meaningful only if the type attribute is set to "identity".

Parameter

A string tht represents the attribute. The valid strings are: "aik-name", "distinguished-name", "dns-name", "email-address", "kerberos-principal", "username", "sip-uri", "tel-uri" and "other". The default value is "username".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get the identity type
my $type = $identifier->identity_type();

# Set the identity type
$identifier->identity_type( "email-address" );
```

ip_address_type( )

Use this method to set or retrieve the value of the ip_address_type attribute. This attribute is meaningful only if the type attribute is set to "ip-address".

Parameter

A string that represents the attribute. The valid values are "IPv4" and "IPv6".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get the IP address type
my $type = $identifier->ip_address_type();

# Set the IP address type
$identifier->ip_address_type( "IPv6" );
```

name( )

Use this method to set or retrieve the value of the name attribute.

The value must be the name of the identifier. Depending on the type, the value can be an IP address, a MAC address, a device name, a device AIK name, an identity name, or an access request name. It is required when type is not "custom".

Parameter

A string that represents the attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $name = $identifier->name();

# Set name
$identifier->name( "John Doe" );
```

type( )

Use this method to set or retrieve the value of the type attribute.

Parameter

A string that represents the attribute. The valid identifier types are: "ip-address", "mac-address", "device", "identity" and "access-request".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get type
my $type = $identifier->type();

# Set type
$identifier->type( "ip-address" );
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::Identifier::Attribute, Infoblox::IFMap::Federation::Subscription, Infoblox::IFMap::Identifier::Definition

---

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::IFMap::Identifier::Attribute - IF-MAP Identifier Attribute object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::IFMap::Identifier->custom_attributes( )
- METHODS
  - name( )
  - value( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Identifier::Attribute - IF-MAP Identifier Attribute object

DESCRIPTION

This object represents an attribute of an IF-MAP identifier. This is used in the context of custom identifiers.

CONSTRUCTOR

```
my $attribute = Infoblox::IFMap::Identifier::Attribute->new(
    name        => $name,           #Required
    value       => $value,          #Required
);
```

MODULE METHODS

The following functions can be applied to an IF-MAP Identifier Attribute object.

Infoblox::IFMap::Identifier->custom_attributes( )

Use this function to set or retrieve the list of attributes of a given identifier. See Infoblox::IFMap::Identifier for parameters and return values.

Example

```
# Add a new attribute
my $attr = Infoblox::IFMap::Identifier::Attribute->new(
                        name => "isbn",
                        value => "978-1565925120"
          );

# Set attributes
$identifier->custom_attributes([$attr]);
```

METHODS

This section describes all the methods that you can use to set or retrieve the attribute values of an IF-MAP Identifier Attribute object.

name( )

Use this method to set or retrieve the value of the name attribute.

Parameter

A string that represents the attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get name
my $name = $attr->name( );

# Change it
$attr->name("isbn");
```

value( )

Use this method to set or retrieve the value of the value attribute.

Parameter

A string that represents the attribute.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get value
my $value = $attr->value( );

# Change it
$attr->value("978-1565925120");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::Identifier

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::IFMap::Identifier::Definition - IF-MAP Custom Identifier Type Definition object

- NAME
- DESCRIPTION
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
  - Infoblox::Session->remove( )
  - Infoblox::Session->search( )
- METHODS
  - attributes( )
  - comment( )
  - display_name( )
  - element_name( )
  - extensible_attributes( )
  - name( )
  - namespace( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Identifier::Definition - IF-MAP Custom Identifier Type (CIT) Definition object

DESCRIPTION

This module provides an interface for manipulating custom identifier definition (sometimes also referred as custom identifier type) objects. These objects are automatically created by importing XSD schemas describing the custom identifiers.

SESSION METHODS

This section describes all the methods in an Infoblox::Session module that you can apply to an IF-MAP identifier definition object.

Infoblox::Session->get( )

Use this method to retrieve the IF-MAP custom identifier definition objects. See Infoblox::Session->get( ) for parameters and return values.

Key References

```
Apply the following attributes to get a specific IF-MAP custom identifier definition object:

name                  - Optional. A string that contains the name of the object. Default is undef.
element_name          - Optional. A string that contains an element name. Default is undef.
display_name          - Optional. A string that contains a display name. Default is undef.
comment               - Optional. A string that contains a comment. Default is undef.
extensible_attributes - Optional. A hash reference that contains extensible attributes. Default is
undef.
```

Example

```
#Get an IF-MAP CIT with the name 'LocationType'
my @retrieved_objs = $session->get(
        object => "Infoblox::IFMap::Identifier::Definition",
        name => "LocationType" );
```

Infoblox::Session->modify( )

Use this method to update the IF-MAP custom identifier definition objects. This method commits the changes applied to the Infoblox::IFMap::Identifier::Definition object. See Infoblox::Session->modify( ) for parameters and return values.

Example

```
#Update the comment of the IF-MAP CIT
$cit->comment("new comment");
#Update the IF-MAP CIT object
my $response = $session->modify( $cit );
```

Infoblox::Session->remove( )

Use this method to remove IF-MAP custom identifier definition objects.

Example

```
#Get the IF-MAP CIT named 'LocationType'
my @retrieved_objs = $session->get(
          object => "Infoblox::IFMap::Identifier::Definition",
          name => "LocationType"
          );

#Retrieve the desired object from the returned list
my $location_cit = $retrieved_objs[0];

#Remove the object
my $response = $session->remove( $location_cit );
```

Infoblox::Session->search( )

Use this method to search for the IF-MAP custom identifier definition objects. See Infoblox::Session->search( ) for parameters and return values.

Key References

```
Apply the following attributes to get a specific IF-MAP custom identifier definition object:

name                  - Optional. A string that contains the name of the object. Default is undef.
Regular expression syntax supported.
element_name          - Optional. A string that contains an element name. Default is undef.
display_name          - Optional. A string that contains a display name. Default is undef. Regular
expression syntax supported.
comment               - Optional. A string that contains a comment. Default is undef. Regular
expression syntax supported.
extensible_attributes - Optional. A hash reference that contains extensible attributes. Default is
undef.
```

Example

```
#Search for IF-MAP CITs named 'Network*'
my @retrieved_objs = $session->search(
         object => "Infoblox::IFMap::Identifier::Definition",
         name => "Network*" );
```

METHODS

attributes( )

Use this method to set or retrieve CIT attributes. If no argument is passed, it returns the current attributes. The attributes are represented by Infoblox::IFMap::Identifier::Definition::Attribute objects. For each attribute one can change only the use_for_name field of the corresponding Infoblox::IFMap::Identifier::Definition::Attribute object. You cannot change the number of attributes or modify the other fields or the Infoblox::IFMap::Identifier::Definition::Attribute object.

Parameter

The valid value is a reference to the array of Infoblox::IFMap::Identifier::Definition::Attribute objects. The objects in the array must have been previously retrieved from the same custom identifier type object.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Get the CIT attributes
my $ref_cit_attributes = $cit->attributes();

#Set the CIT attributes
$cit->attributes($attrs);
```

comment( )

Use this method to set or retrieve the value of the comment attribute. If no argument is passed, the method returns the current value of the comment attribute.

Parameter

The valid value is a string that contains the desired comment. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Get the CIT comment
my $comment = $cit->comment();

#Change the CIT comment
$cit->comment("New Comment");
```

display_name( )

Use this method to set or retrieve the value of the display name attribute. If no argument is passed, the method returns the current value of the display name attribute. The display name is the user-friendly name displayed in the WebUI.

Parameter

A string that contains the display name. The initial value of the display name is set when the IF-MAP custom identifier type object is imported.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Get CIT display name
my $display_name = $cit->display_name();

#Change CIT display name
$cit->display_name("New Name");
```

element_name( )

Use this method to retrieve the value of the element name attribute. This attribute is read-only. It is set by the server, so it has a value only when the server reads the existing Infoblox::IFMap::Identifier::Definition object.

Parameter

none

Returns

The method returns the element name attribute of the IF-MAP CIT object.

Examples

```
#Get the IF-MAP CIT element name
my $name = $cit->element_name();
```

extensible_attributes( )

Use this method to set or retrieve extensible attributes for this object. If no argument is passed, it returns the list of current attributes.

Parameter

The valid value is a hash reference that contains the names of extensible attributes and their associated values. The names must be the names of existing Infoblox::Grid::ExtensibleAttributeDef objects in the database. For email, URL, and string types, the value is a string with a maximum of 256 characters. For a list, the value is a string with a maximum of 64 characters. For an integer, the value is an integer from -2147483648 through 2147483647. For a date, the value is the number of seconds that have elapsed since January 1st, 1970 UTC. The default value is undefined.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Get the CIT extensible attributes
my $ref_extensible_attributes = $cit->extensible_attributes();
```

```
#Modify the CIT extensible attributes
$cit->extensible_attributes({ 'Administrator' => [ 'David', 'Peter', 'Tom' ] });
```

name( )

Use this method to retrieve the value of the name attribute. This attribute is read-only. It is set by the server, so it has a value only when the server reads the existing Infoblox::IFMap::Identifier::Definition object.

Parameter

none

Returns

The method returns the name of the IF-MAP custom identifier type.

Examples

```
#Get the IF-MAP CIT name
my $name = $cit->name();
```

namespace( )

Use this method to retrieve the value of the namespace attribute. This attribute is read-only. It is set by the server, so it has a value only when the server reads the existing Infoblox::IFMap::Identifier::Definition object.

Parameter

none

Returns

The method returns the namespace attribute of the IF-MAP CIT object.

Examples

```
#Get the IF-MAP CIT namespace
my $namespace = $cit->namespace();
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and remove. This sample also includes error handling for the operations.

#Preparations

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
   master   => "192.168.1.2", #appliance host ip
   username => "admin",       #appliance user login
   password => "infoblox"     #appliance user password
);

unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Import the IF-MAP identifier definition from the XSD schema file

```
my $res = $session->import_data(
   type => "ifmap_cit_schema",
   path => "../bin/schemas/schema_ok.xsd",
   display_name => "location_1",
);

unless ($res) {
   die("Import of schema file failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
```

```
print "Imported IFMap schema successfully\n";

#Get the IF-MAP custom identifier definition object
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Identifier::Definition",
    display_name => "location_1",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get IFMap identifier definition object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get IFMap identifier definition object found at least 1 matching entry\n";
```

#Modify the IF-MAP custom identifier definition object

```
#Modify display_name
unless($object->display_name("location_17")){
       die("Setting display_name attribute for IFMap custom identifier definition failed: ",
            Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Modified object display_name field successfully.\n";

#Modify attributes
my $ref_attributes = $object->attributes();
if(ref($ref_attributes) eq 'ARRAY' && scalar(@$ref_attributes)>0){
       unless($$ref_attributes[0]->use_for_name("true")){
           die("Modifying second attribute failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
       unless($object->attributes($ref_attributes)){
           die("Modifying attributes field failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
       }

}else{
       print "Object has no attributes\n";
}

#Submit object to the appliance
$session->modify($object)
       or die("Modify IFMap CIT failed: ",
              $session->status_code() . ":" .  $session->status_detail());
print "Modified object on the appliance successfully.\n";
```

#Remove object from the appliance

```
@retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Identifier::Definition",
    display_name => "location_17",
);
$object = $retrieved_objs[0];
$session->remove($object)
       or die("Removing object from database failed: ",
              $session->status_code() . ":" . $session->status_detail());
print "Removed object from the database successfully.\n";
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::AuthRule, Infoblox::IFMap::Identifier::Definition::Attribute, Infoblox::Session,Infoblox::Session->get( ),Infoblox::Session->import_data( ),Infoblox::Session->modify( ),Infoblox::Session->remove( ),Infoblox::Session->search( )

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::IFMap::Identifier::Definition::Attribute - IF-MAP custom identifier definition attribute object.

- NAME
- DESCRIPTION
- MODULE METHODS
  - Infoblox::IFMap::Identifier::Definition->attributes( )
- METHODS
  - name( )
  - required( )
  - restrictions( )
  - type( )
  - use_for_name( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IFMap::Identifier::Definition::Attribute - IF-MAP custom identifier definition attribute object.

DESCRIPTION

This module provides an interface for manipulating the attribute objects of the IF-MAP custom identifier type (IF-MAP CIT) objects.

MODULE METHODS

Infoblox::IFMap::Identifier::Definition->attributes( )

Use this method to retrieve or modify IF-MAP CIT attributes. See Infoblox::IFMap::Identifier::Definition).

Examples

```
#Get the CIT attributes
my $ref_cit_attributes = $cit->attributes();

#Modify the CIT attributes
$$ref_cit_attributes[0]->use_for_name("true");
$cit->attributes($ref_cit_attributes);
```

METHODS

name( )

Use this method to retrieve the value of the name attribute. This attribute is read-only. It is set by the server, so it has a value only when the server reads the existing Infoblox::IFMap::Identifier::Definition object.

Parameter

none

Returns

The method returns the name of the Infoblox::IFMap::Identifier::Definition::Attribute object.

Examples

```
#Get the first attribute of the IF-MAP CIT
my $attribute = $cit->attributes()->[0];

#Get the name of the IF-MAP CIT attribute
my $name = $attribute->name();
```

required( )

Use this method to retrieve the value of the required attribute. This attribute is read-only. The attribute is set by server, so it has a value only when the server reads the Infoblox::IFMap::Identifier::Definition::Attribute object.

Parameter

none

Returns

The method returns the 'required' field of the Infoblox::IFMap::Identifier::Definition::Attribute object.

Examples

```
#Get the required flag of the IF-MAP CIT attribute
my $required_flag = $attribute->required();
```

restrictions( )

Use this method to retrieve the value of the restrictions attribute. This attribute is read-only. The attribute is set by server, so it has a value only when the server reads the existing Infoblox::IFMap::Identifier::Definition::Attribute object. The interpretation of this field depends on the value of "type" field. If type field is set to "enum" the restriction field contains a list of allowed enum values, if type field is set to "string" the restriction field contains the regular expression that the string should match to.

Parameter

none

Returns

The method returns the restrictions attribute of the Infoblox::IFMap::Identifier::Definition::Attribute object.

Examples

```
#Get the restrictions of the IF-MAP CIT attribute
my $restrictions = $attribute->restrictions();
```

type( )

Use this method to retrieve the value of the type attribute. This attribute is read-only. It is set by the server, so it has a value only when the server reads the existing Infoblox::IFMap::Identifier::Definition::Attribute object.

Parameter

none

Returns

The method returns the type of the Infoblox::IFMap::Identifier::Definition::Attribute object. Possible returned values for type are int, string, or enum.

Examples

```
#Get the attribute type of the IF-MAP CIT
my $type = $attribute->type();
```

use_for_name( )

Use this method to set or retrieve the value of the use_for_name attribute. If no argument is passed, it returns the current value.

Parameter

Valid values are "true" or "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Examples

```
#Get the use_for_name flag of the IF-MAP CIT attribute
my $use_for_name_flag = $attribute->use_for_name();

#Set the use_for_name flag of the IF-MAP CIT attribute
$attribute->use_for_name("true");
```

SAMPLE CODE

The following sample code demonstrates the different functions that can be applied to an object, such as add, search, modify, and

remove. This sample also includes error handling for the operations.

#Preparations #PROGRAM STARTS: Include all the modules that will be used use strict; use Infoblox;

```
#Create a session to the Infoblox appliance

my $session = Infoblox::Session->new(
   master   => "192.168.1.2", #appliance host ip
   username => "admin",       #appliance user login
   password => "infoblox"     #appliance user password
);

unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";
```

#Import the IF-MAP identifier definition from the XSD schema file my $res = $session->import_data(type => "ifmap_cit_schema", path => "../bin/schemas/schema_ok.xsd", display_name => "location_1",);

```
unless ($res) {
   die("Import of schema file failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Imported IFMap schema successfully\n";

#Get the IF-MAP custom identifier definition object
my @retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Identifier::Definition",
    display_name => "location_1",
);
my $object = $retrieved_objs[0];
unless ($object) {
       die("Get IFMap identifier definition object failed: ",
            $session->status_code() . ":" . $session->status_detail());
}
print "Get IFMap identifier definition object found at least 1 matching entry\n";
```

#Modify the IF-MAP custom identifier definition object

```
#Modify attributes
my $ref_attributes = $object->attributes();
if(ref($ref_attributes) eq 'ARRAY' && scalar(@$ref_attributes)>0){
       unless($$ref_attributes[0]->use_for_name("true")){
           die("Modifying second attribute failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
       }
       unless($object->attributes($ref_attributes)){
           die("Modifying attributes field failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
       }

}else{
       print "Object has no attributes\n";
}

#Submit the object to the appliance
$session->modify($object)
       or die("Modify IFMap CIT failed: ",
              $session->status_code() . ":" .  $session->status_detail());
print "Modified object on the appliance successfully.\n";
```

#Remove the object from the appliance

```
@retrieved_objs = $session->get(
    object => "Infoblox::IFMap::Identifier::Definition",
    display_name => "location_17",
);
$object = $retrieved_objs[0];
$session->remove($object)
       or die("Removing object from database failed: ",
              $session->status_code() . ":" . $session->status_detail());
print "Removed object from the database successfully.\n";
```

####PROGRAM ENDS####

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IFMap::AuthRule, Infoblox::IFMap::Identifier::Definition, Infoblox::Session, Infoblox::Session->get( ), Infoblox::Session->modify( ), Infoblox::Session->remove( ), Infoblox::Session->search( )

COPYRIGHT

Copyright (c) 1999-2010 Infoblox Inc.

Infoblox::IPAM::Address - IPAM address object

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->remove( )
- METHODS
  - dhcp_client_identifier( )
  - discovered_name( )
  - duid( )
  - discoverer( )
  - extensible_attributes( )
  - first_discovered( )
  - ip_address( )
  - is_conflict( )
  - last_discovered( )
  - lease_state( )
  - mac_address( )
  - name( )
  - netbios_name( )
  - network_component_description( )
  - network_component_ip( )
  - network_component_name( )
  - network_component_port_description( )
  - network_component_port_name( )
  - network_component_port_number( )
  - network_component_type( )
  - network_view( )
  - os( )
  - port_duplex( )
  - port_link_status( )
  - port_speed( )
  - port_status( )
  - port_vlan_description( )
  - port_vlan_name( )
  - port_vlan_number( )
  - status( )
  - types( )
  - usage( )
  - username( )
  - v_cluster( )
  - v_datacenter( )
  - v_host( )
  - v_name( )
  - v_netadapter( )
  - v_switch( )
  - v_type( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IPAM::Address - IPAM address object

DESCRIPTION

The IPAM address object is used to view IPv4 and IPv6 address management information in an Infoblox appliance.

CONSTRUCTOR

The Infoblox::IPAM::Address object is a read-only object and does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to an IPAM address object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Note that for IPv4 addresses, both used and unused addresses are returned unless 'status' is specified; but IPv6 addresses will only return used addresses.

Key Reference

Apply the following attributes to get the IPAM address object(s). Note that at least one of the following is required: address, start_addr, end_addr, or network.

address - Optional. An IPv4 or IPv6 address. Get IPAM information for this address. All other optional arguments are ignored, except for network_view, if 'address' is present.

network - Optional. The IPv4 or IPv6 network, in CIDR notation (e.g 1.0.0.0/8 or 2000::/64), where the IPAM address is located.

network_view - Optional. The network view in which the network is located. The default is the system-defined default network view.

start_addr - Optional. An IPv4 or IPv6 address. Start address of an IPAM address scope. If the start address is not in the specified network, an error is returned.

end_addr - Optional. An IPv4 or IPv6 address. End address of an IPAM address scope. If the end address is not in the specified network, an error is returned.

status - Optional. The filter type of the IPAM address. Valid values are "used", "unused" and "all". Default value is "all". This argument applies to IPv4 addresses only. When the value is returned ", Unmanaged" or ", Conflict" might be appended to the status member when applicable.

type - Optional. This is exactly the same as the 'status' argument, and is being deprecated in favor of the 'status' argument. This argument applies to IPv4 addresses only.

extensible_attributes - Optional. A hash reference containing extensible attributes.

v_cluster - Optional. Discovered VMware cluster name.

v_datacenter - Optional. Discovered VMware datacenter name.

v_host - Optional. Discovered VMware server name.

v_name - Optional. Discovered VMware host or virtual machine name.

v_netadapter - Optional. Discovered VMware physical network adapter name.

v_switch - Optional. Discovered VMware virtual switch name.

v_type - Optional. Discovered VMware entity type (possible values: "VirtualMachine", "HostSystem", "VirtualCenter").

Example

```
# case 1
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'network'  => '10.0.0.0/8');
```

Since there are no start and end addresses specified, the appliance returns a maximum of 2000 IP addresses, which is the maximum page size of an administrator. If the total number of IP addresses in the network is less than 2000, then it returns all IP addresses.

```
# case 2
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'network'    => '10.0.0.0/8',
                                  'start_addr' => '10.0.0.100');
```

This call returns a maximum of 2000 IPAM addresses starting from 10.0.0.100.

```
# case 3
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'network'    => '10.0.0.0/8',
                                  'start_addr' => '10.0.0.100'
                                  'end_addr'   => '10.0.0.200');
```

This call returns 101 IPAM addresses from 10.0.0.100 to 10.0.0.200.

```
# case 4
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'address'    => '10.0.0.10');
```

This call returns the IPAM address object for 10.0.0.10.

```
# case 5
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'network'  => '10.0.0.0/8',
                                  'start_addr' => '10.0.0.100',
                                  'end_addr'  => '10.0.0.200',
                                  'status'          => 'used' );
```

This call returns used IPAM addresses from 10.0.0.100 to 10.0.0.200. Note that the filter is applied to the specified scope. Therefore, the number of returned objects could be less than 101, if only a few IP addresses are used in this scope.

```
# case 6
my @ipam_objects = $session->get( 'object        => 'Infoblox::IPAM::Address',
                                  'network'      => '10.0.0.0/8',
                                  'network_view' => 'my_network_view');
```

Same as case 1, but this uses the 'my_network_view' network view rather than the system-defined default view.

```
# case 7
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'address'    => '10.0.0.5');
# case 8
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'network'  => '2000::/64');
```

Since there are no start and end addresses specified, the appliance returns a maximum of 2000 used IP addresses, which is the maximum page size of an administrator. If the total number of IP addresses in the network is less than 2000, then it returns all IP addresses.

```
# case 9
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'network'    => '2000::/64',
                                  'start_addr' => '2000::11');
```

This call returns a maximum of 2000 used IPv6 addresses starting from 2000::11.

```
# case 10
my @ipam_objects = $session->get( 'object'     => 'Infoblox::IPAM::Address',
                                  'network'    => '2000::/64',
                                  'start_addr' => '2000::11',
                                  'end_addr'   => '2000::31');
```

This call returns used IPv6addresses from '2000::11' to '2000::31'.

Infoblox::Session->remove( )

Use this method to remove an object from the Infoblox appliance. See Infoblox::Session->remove() for parameters and return values.

To remove a specific object, first use `get()` to retrieve the specific object, and then submit this object for removal.

Example

```
# Get the object under the same name
my @result_array = $session->get(
      object        => "Infoblox::IPAM::Address" ,
      network       => "2001::/64",
);

# find the desired object from the retrieved list.
my $desired_ipam_address = $result_array[0];

# Submit for removal
my $response = $session->remove( $desired_ipam_address );
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of an IPAM address object.

dhcp_client_identifier( )

Use this method to retrieve the DHCP client identifier of an IPAM address object. This method applies to IPv4 addresses only.

Parameter

none

Returns

The method returns the DHCP client identifier attribute value.

Example

```
#Get DHCP client identifier
my $identifier = $ipam_address->dhcp_client_identifier();
```

discovered_name( )

Use this method to retrieve the discovered name of an IPAM address object.

Parameter

none

Returns

The method returns the discovered name.

Example

```
#Get discovered_name
my $discovered_name = $ipam_address->discovered_name();
```

duid( )

Use this method to retrieve the DHCPv6 Unique Identifier (DUID) of an IPAM address object.

Parameter

none

Returns

The method returns the DUID value.

Example

```
#Get mac address
my $duid = $ipam_address->duid();
```

discoverer( )

Use this method to retrieve the discoverer of an IPAM address object.

Parameter

none

Returns

The method returns the discoverer value.

Example

```
#Get discoverer
my $discoverer = $ipam_address->discoverer();
```

extensible_attributes( )

Use this method to retrieve the extensible attributes associated with an IPAM address object.

Parameter

none

Returns

The method returns a reference to a hash containing the extensible attributes.

Example

```
#Get extensible attributes
my $ref_extensible_attributes = $ipam_address->extensible_attributes();
```

first_discovered( )

Use this method to retrieve the date and time that the IPAM address object was first discovered.

Parameter

none

Returns

The method returns the date and time that the IPAM address object was fisrt discovered.

Example

```
#Get first_discovered
my $first_discovered = $ipam_address->first_discovered();
```

ip_address( )

Use this method to retrieve the IP addresses of an IPAM address object.

Parameter

none

Returns

The method returns the IP address value. The value can be an IPv4 or IPv6 address.

Example

```
#Get IP address
my $ip_address = $ipam_address->ip_address();
```

is_conflict( )

Use this method to find out whether this IPAM address is in conflict with another IPAM address in the system. This method applies to IPv4 addresses only.

Parameter

none

Returns

The method returns true if the IPAM address is in conflict with another in the system, false otherwise.

Example

```
#Get is_conflict
my $conflict = $ipam_address->is_conflict();
```

last_discovered( )

Use this method to retrieve the last discovered time of an IPAM address object.

Parameter

none

Returns

The method returns the last discovered value.

Example

```
#Get last discovered
my $last_discovered = $ipam_address->last_discovered();
```

lease_state( )

Use this method to retrieve the lease state of an IPAM address object. This method applies to IPv4 addresses only.

Parameter

none

Returns

The method returns the lease state value. A valid state is any of the following string values: free, backup, active, expired, released, abandoned, and reset.

Example

```
#Get lease state
my $lease_state = $ipam_address->lease_state();
```

mac_address( )

Use this method to retrieve the MAC address of an IPAM address object.

Parameter

none

Returns

The method returns the MAC address value.

Example

```
#Get mac address
my $mac_address = $ipam_address->mac_address();
```

name( )

Use this method to retrieve the name of an IPAM address object.

Parameter

none

Returns

The method returns the name value.

Example

```
#Get name
my $name = $ipam_address->name();
```

netbios_name( )

Use this method to retrieve the NETBIOS name of an IPAM address object.

Parameter

none

Returns

The method returns the NETBIOS name value.

Example

```
#Get netbios name
my $netbios_name = $ipam_address->netbios_name();
```

network_component_description( )

Use this method to retrieve the description of the network device that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device description.

Example

```
#Get network_component_description
my $network_component_description = $ipam_address->network_component_description();
```

network_component_ip( )

Use this method to retrieve the IP address of the network device that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device IP address.

Example

```
#Get network_component_ip
my $network_component_ip = $ipam_address->network_component_ip();
```

network_component_name( )

Use this method to retrieve the name of the network device that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device name.

Example

```
#Get network_component_name
my $network_component_name = $ipam_address->network_component_name();
```

network_component_port_description( )

Use this method to retrieve the description of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device port description.

Example

```
#Get network_component_port_description
my $network_component_port_description = $ipam_address->network_component_port_description();
```

network_component_port_name( )

Use this method to retrieve the name of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device port name.

Example

```
#Get network_component_port_name
my $network_component_port_name = $ipam_address->network_component_port_name();
```

network_component_port_number( )

Use this method to retrieve the number of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device port number in the range 0-9999.

Example

```
#Get network_component_port_number
my $network_component_port_number = $ipam_address->network_component_port_number();
```

network_component_type( )

Use this method to retrieve the type of the network device that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device type.

Example

```
#Get network_component_type
my $network_component_type = $ipam_address->network_component_type();
```

network_view( )

Use this method to retrieve the network view name of an IPAM address object.

Parameter

none

Returns

The method returns the network view name.

Example

```
#Get network view
my $nview_name = $ipam_address->network_view();
```

os( )

Use this method to retrieve the OS of an IPAM address object.

Parameter

none

Returns

The method returns the OS value.

Example

```
#Get os
my $os = $ipam_address->os();
```

port_duplex( )

Use this method to retrieve the duplex setting of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the port duplex setting. Returned values, if any, are one of the following:

```
Full
Half
```

Example

```
#Get port_duplex
my $port_duplex = $ipam_address->port_duplex();
```

port_link_status( )

Use this method to retrieve the link status of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the port link status of the network device. Returned values, if any, are one of the following:

```
Connected
Not Connected
Unknown
```

Example

```
#Get port_link_status
my $port_link_status = $ipam_address->port_link_status();
```

port_speed( )

Use this method to retrieve the speed of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the port speed. Returned values, if any, are one of the following:

```
10M
100M
1G
10G
100G
Unknown
```

Example

```
#Get port_speed
my $port_speed = $ipam_address->port_speed();
```

port_status( )

Use this method to retrieve the status of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device port status. Returned values, if any, are one of the following:

```
Up
Down
Unknown
```

Example

```
#Get port_status
my $port_status = $ipam_address->port_status();
```

port_vlan_description( )

Use this method to retrieve the VLAN description of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device port VLAN description.

Example

```
#Get port_vlan_description
my $port_vlan_description = $ipam_address->port_vlan_description();
```

port_vlan_name( )

Use this method to retrieve the VLAN name of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device port VLAN name.

Example

```
#Get port_vlan_name
my $port_vlan_name = $ipam_address->port_vlan_name();
```

port_vlan_number( )

Use this method to retrieve the VLAN number of the network device port that is connected to the IPAM address object.

Parameter

none

Returns

The method returns the network device port VLAN number in the range 0-9999.

Example

```
#Get port_vlan_number
my $port_vlan_number = $ipam_address->port_vlan_number();
```

status( )

Use this method to retrieve the status of an IPAM address object.

Parameter

none

Returns

The method returns the status value.

Example

```
#Get status
my $status = $ipam_address->status();
```

types( )

Use this method to retrieve the type of an IPAM address object.

Parameter

none

Returns

The method returns the type value. A valid value is a comma-separated list that contains a subset of the following values: A, FA, RESERVATION, RANGE, LEASE, UNMANAGED, HOST, BULKHOST, PTR, NETWORK, and BROADCAST. Note that in the subset of values, "FA" refers to "fixed address".

Example

```
#Get types
my $types= $ipam_address->types();
```

usage( )

Use this method to retrieve the usage of an IPAM address object.

Parameter

none

Returns

The method returns the usage value. A valid value is a comma-separated list that contains a subset of the following values: DNS, DHCP.

Example

```
#Get usage
my $usage= $ipam_address->usage();
```

username( )

Use this method to retrieve the user name of an IPAM address object. This method applies to IPv4 addresses only.

Parameter

none

Returns

The method returns the user name.

Example

```
#Get username
my $username= $ipam_address->username();
```

v_cluster( )

Use this method to retrieve the name of the VMware cluster associated with the IPAM Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware cluster name. The default value is an empty string.

Example

```
#Get v_cluster
my $v_cluster = $ipam_address->v_cluster();
```

v_datacenter( )

Use this method to retrieve the name of the VMware datacenter associated with the IPAM Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware datacenter name. The default value is an empty string.

Example

```
#Get v_datacenter
my $v_datacenter = $ipam_address->v_datacenter();
```

v_host( )

Use this method to retrieve the name of the VMware host associated with the IPAM Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware host name. The default value is an empty string.

Example

```
#Get v_host
my $v_host = $ipam_address->v_host();
```

v_name( )

Use this method to retrieve the name of the VMware entity associated with the IPAM Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware entity name. The default value is an empty string.

Example

```
#Get v_name
my $v_name = $ipam_address->v_name();
```

v_netadapter( )

Use this method to retrieve the name of the physical network adapter through which the VMware entity is connected to the appliance. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the network adapter name. The default value is an empty string.

Example

```
#Get v_netadapter
my $v_netadapter = $ipam_address->v_netadapter();
```

v_switch( )

Use this method to retrieve the name of the virtual switch through which the VMware entity is connected to the appliance. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the virtual switch name. The default value is an empty string.

Example

```
#Get v_switch
my $v_switch = $ipam_address->v_switch();
```

v_type( )

Use this method to retrieve the type of VMware entity associated with the IPAM Address object. This method is read-only.

Parameter

none

Returns

This method returns a string that contains the VMware entity type. The default value is an empty string. The returned value, if any, can be one of the following:

```
VirtualMachine
HostSystem
VirtualCenter
```

Example

```
#Get v_type
my $v_type = $ipam_address->v_type();
```

SAMPLE CODE

The following sample code demonstrates the get session method on an IPAM address object.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

my $host_ip = "192.168.1.2";
my $host_name = "localhost";
my $node_ip = "192.168.1.2";

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
               master   => $host_ip,
               username => "admin",
               password => "infoblox"
               );

unless ($session) {
       die("Construct session failed: ",
               Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $memberdhcp = Infoblox::DHCP::Member->new(
          name     => $host_name,
          ipv4addr => $node_ip,
     );

#Create a large network object
my $network_large = Infoblox::DHCP::Network->new(
                                          "network" => "10.0.0.0/8",
                                          "members" => [$memberdhcp],
                                         );

$session->add($network_large)
   or die("Add network failed: " . $session->status_code() . ": " . $session->status_detail());

#Create another member
my $new_member = Infoblox::Grid::Member->new(
       grid      => "0",
       name      => "foo.com",
       ipv4addr  => "3.0.0.2",
       mask      => "255.255.255.0",
       gateway   => "3.0.0.1",
     );
$session->add($new_member)
   or die("Add grid member failed: " . $session->status_code() . ": " . $session->status_detail());

my $memberdhcp2 = Infoblox::DHCP::Member->new(
          name     => "foo.com",
          ipv4addr => "3.0.0.2",
     );

#Create a network view
my $nview = Infoblox::DHCP::View->new(
                                  "name" => "my_network_view",
                                 );
$session->add($nview)
   or die("Add network view failed: " . $session->status_code() . ": " . $session->status_detail());

#Create a DNS view associated with the network view
my $view = Infoblox::DNS::View->new(
  "name" => "my_dns_view",
  "network_view" => $nview,
);
$session->add($view)
   or die("Add DNS view failed: " . $session->status_code() . ": " . $session->status_detail());

#Create a DNS Zone in a non-default DNS view
my $zone = Infoblox::DNS::Zone->new(
  "name" => "test.com",
  "views" => [$view],
);
$session->add($zone)
   or die("Add DNS Zone failed: " . $session->status_code() . ": " . $session->status_detail());

#Create a small IPv4 network object with a non-default network view
```

```
my $network_small = Infoblox::DHCP::Network->new(
                                          "network" => "20.0.0.0/24",
                                          "members" => [$memberdhcp2],
                                          "network_view" => $nview,
                                         );

$session->add($network_small)
   or die("Add network failed: " . $session->status_code() . ": " . $session->status_detail());

#Create an IPv6 network object with a non-default network view
my $ipv6_network = Infoblox::DHCP::IPv6Network->new(
    "network" => "2001::/64",
    "network_view" => $nview,
);
$session->add($ipv6_network)
   or die("Add IPv6 network failed: " . $session->status_code() . ": " . $session->status_detail());

#Construct an IPv4 DHCP range object
my $dhcp_range = Infoblox::DHCP::Range->new(
        start_addr              => "10.0.0.1",
        end_addr                => "10.0.0.10",
        authority               => "true",
        bootfile                => "bootfile1.com",
        bootserver              => "someserver.com",
        comment                 => "add range",
        ddns_generate_hostname  => "true",
        deny_all_clients        => "true",
        deny_bootp              => "true",
        disable                 => "true",
        enable_ddns             => "true",
        member                  => $memberdhcp,
        nextserver              => "3.3.3.3",
        pxe_lease_time          => "3600",
        recycle_leases          => "true",
);

$session->add($dhcp_range)
   or die("Add DHCP range failed: " . $session->status_code() . ": " . $session->status_detail());

#Construct an IPv4 DHCP fixed address object
my $fixed_addr = Infoblox::DHCP::FixedAddr->new(
      "ipv4addr"       => "10.0.0.15",
      "mac"            => "11:11:11:11:11:11",
      "dhcp_client_identifier" => "",
      "client_identifier_prepend_zero"   => "false",
      "match_client"   => "MAC",
      "network"        => "10.0.0.0/8",
      "bootfile"       => "bootfile",
      "bootserver"     => "bootserver.com",
      "comment"        => "add fixed address",
      "deny_bootp"     => "false",
      "disable"        => "false",
      "dynamic"        => "false",
      "enable_ddns"    => "false",
      "nextserver"     => "3.3.3.3",
      "pxe_lease_time" => "3600",
);

$session->add($fixed_addr)
   or die("Add DHCP range failed: " . $session->status_code() . ": " . $session->status_detail());

#Construct IPv6 Host and AAAA record objects in a non-default DNS view
my $host = Infoblox::DNS::Host->new(
      "name"           => "host1.test.com",
      "ipv6addrs"      => ["2001::1", "2001::2", "2001::5"],
      "views"          => [$view],
);

$session->add($host)
   or die("Add host failed: " . $session->status_code() . ": " . $session->status_detail());

my $aaaa = Infoblox::DNS::Record::AAAA->new(
      "name"           => "myhost1.test.com",
      "ipv6addr"       => "2001::1",
      "views"          => [$view],
);
```

```
$session->add($aaaa)
   or die("Add AAAA record failed: " . $session->status_code() . ": " . $session->status_detail());

   # Case 1: Get IPAM address objects using 'network'
   # This should return 2000 IPAM addresses
   my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                    'network'    => '10.0.0.0/8');
   unless( scalar(@ipam_objects) == 2000) {
       die("Returned IPAM addresses number is not correct");
   }
   print("Returned IPAM addresses number correctly\n");

   # Case 2: Get IPAM address objects using 'start_addr'
   # This should return 2000 IPAM addresses
   my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                    'network'    => '10.0.0.0/8',
                                    'start_addr' => '10.0.0.10');
   unless( scalar(@ipam_objects) == 2000) {
       die("Returned IPAM addresses number is not correct");
   }
   print("Returned IPAM addresses number correctly\n");

   # Case 3: Get IPAM address objects using 'start_addr' and 'end_addr'
      # This should return the specified number of IPAM addresses
   my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                    'network'    => '10.0.0.0/8',
                                    'start_addr' => '10.0.0.10',
                                    'end_addr'   => '10.0.0.100');

   unless( scalar(@ipam_objects) == 91 ) {
       die("Returned IPAM addresses number is not correct");
   }
   print("Returned IPAM addresses number correctly\n");

   # Case 4: Get IPAM address objects using 'address'
   # This should return one IPAM address
   my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                    'address'    => '10.0.0.10');
   unless( scalar(@ipam_objects) == 1 ) {
       die("Returned IPAM addresses number is not correct");
   }
   print("Returned IPAM addresses number correctly\n");

  # Case 5: Get IPAM address objects using a 'start_addr' that is close to the end of the network
  # This should return IPAM addresses from the start_addr to the end of the  network
   my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                    'network'    => '10.0.0.0/8',
                                    'start_addr' => '10.255.255.100');

   unless( scalar(@ipam_objects) == 156 ) {
       die("Returned IPAM addresses number is not correct");
   }
   print("Returned IPAM addresses number correctly\n");

   # Case 6: Get IPAM address objects using 'start_addr' and 'end_addr'
      # but the total number of IP addresses in this scope is more than 2000.
      # This should return 2000 IPAM addresses starting with the start_addr.
   my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                    'network'    => '10.0.0.0/8',
                                    'start_addr' => '10.0.0.100',
                                    'end_addr'   => '10.255.255.100');

   unless( scalar(@ipam_objects) == 2000 ) {
       die("Returned IPAM addresses number is not correct");
   }
   print("Returned IPAM addresses number correctly\n");

  # Case 7: Get IPAM address objects from a small network.
  # This should return all IP addresses in this network.
   my @ipam_objects = $session->get('object'       => 'Infoblox::IPAM::Address',
                                    'network'      => '20.0.0.0/24',
                                    'network_view' => 'my_network_view');

   unless( scalar(@ipam_objects) == 256 ) {
       die("Returned IPAM addresses number is not correct");
   }
   print("Returned IPAM addresses number correctly\n");
```

```
# Case 8: Get IPAM address objects using 'type = used'
   # This should return a total of 11 IP addresses in this network
   # which includes 10 in the range and 1 fixed address
my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                 'network'    => '10.0.0.0/8',
                                 'start_addr' => '10.0.0.0',
                                 'end_addr'   => '10.0.0.100',
                                 'type'       => 'used');

unless( scalar(@ipam_objects) == 11 ) {
    die("Returned IPAM addresses number is not correct");
}
print("Returned IPAM addresses number correctly\n");

# Case 9: Get IPAM address objects using 'type = unused'
   # This should return a total of 90 (101 - 11 = 90) IP addresses in this network
   # which excludes 10 in the range and 1 fixed address
my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                 'network'    => '10.0.0.0/8',
                                 'start_addr' => '10.0.0.0',
                                 'end_addr'   => '10.0.0.100',
                                 'type'       => 'unused');

unless( scalar(@ipam_objects) == 90 ) {
    die("Returned IPAM addresses number is not correct");
}
print("Returned IPAM addresses number correctly\n");

# Case 10: Get IPAM address objects using 'address'
   # This should return one IPAM address object for later tests
my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                 'address'    => '10.0.0.15');
unless( scalar(@ipam_objects) == 1 ) {
    die("Returned IPAM addresses number is not correct");
}
print("Returned IPAM addresses number correctly\n");

my $ipam_object = $ipam_objects[0];

unless($ipam_object->status() eq "Used") {
    die("Returned IPAM address object's status is wrong");
}
print("Returned IPAM address object's status correctly\n");

unless($ipam_object->usage() eq "DHCP") {
    die("Returned IPAM address object's usage is wrong");
}
print("Returned IPAM address object's usage correctly\n");

unless($ipam_object->mac_address() eq $fixed_addr->mac()) {
    die("Returned IPAM address object's mac address is wrong");
}
print("Returned IPAM address object's mac address correctly\n");

unless($ipam_object->dhcp_client_identifier() eq $fixed_addr->dhcp_client_identifier()) {
    die("Returned IPAM address object's dhcp client identifier is wrong");
}
print("Returned IPAM address object's dhcp client identifier correctly\n");

# Case 11: Get IPv6 IPAM address objects using 'network'
# This should return 3 used IPAM addresses
my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                 'network'    => '2001::/64',
                                 'network_view' => 'my_network_view');
unless( scalar(@ipam_objects) == 3) {
    die("Returned IPAM addresses number is not correct");
}
print("Returned IPAM addresses number correctly\n");

# Case 12: Get IPv6 IPAM address objects using 'start_addr'
# This should return 2 used IPAM addresses (2001::2, 2001::5)
my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                 'network'    => '2001::/64',
                                 'start_addr' => '2001::2',
                                 'network_view' => 'my_network_view');
unless( scalar(@ipam_objects) == 2) {
    die("Returned IPAM addresses number is not correct");
}
```

```
  print("Returned IPAM addresses number correctly\n");

  # Case 13: Get IPv6 IPAM address objects using 'start_addr' and 'end_addr'
     # This should return 2 used IPAM addresses
  my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                   'network'    => '2001::/64',
                                   'start_addr' => '2001::1',
                                   'end_addr'   => '2001::3',
                                   'network_view' => 'my_network_view');
  unless( scalar(@ipam_objects) == 2 ) {
      die("Returned IPAM addresses number is not correct");
  }
  print("Returned IPAM addresses number correctly\n");

  # Case 14: Get IPv6 IPAM address objects using 'address'
  # This should return one IPAM address
  my @ipam_objects = $session->get('object'     => 'Infoblox::IPAM::Address',
                                   'address'    => '2001::5',
                                   'network_view' => 'my_network_view');
  unless( scalar(@ipam_objects) == 1 ) {
      die("Returned IPAM addresses number is not correct");
  }
  print("Returned IPAM addresses number correctly\n");

  my $ipam_object = $ipam_objects[0];

  unless($ipam_object->ip_address() eq "2001::5") {
      die("Returned IPAM address object's address is wrong");
  }
  print("Returned IPAM address object's address correctly\n");

  unless($ipam_object->status() eq "Used") {
      die("Returned IPAM address object's status is wrong");
  }
  print("Returned IPAM address object's status correctly\n");

  unless($ipam_object->usage() eq "DNS") {
      die("Returned IPAM address object's usage is wrong");
  }
  print("Returned IPAM address object's usage correctly\n");

# Remove IPAM Address object
unless($session->remove($ipam_object)) {
        die("Remove IPAM Address object failed");
}
print "Removed IPAM Address correctly\n";

my @retrieved_objs = $session->get(
     object       => "Infoblox::DHCP::Network" ,
     network      => "10.0.0.0/8",
);
my $network = $retrieved_objs[0];

unless($session->remove($network)) {
        die("Remove Network object failed");
}
print "Removed network correctly\n";

@retrieved_objs = $session->get(
     object       => "Infoblox::DHCP::Network" ,
     network      => "20.0.0.0/24",
     network_view => "my_network_view",
);

$network = $retrieved_objs[0];

unless($session->remove($network)) {
        die("Remove Network object failed");
}
print "Removed network correctly\n";

$view = $session->get(
    object  => "Infoblox::DNS::View",
    name    => "my_dns_view",
);
unless($session->remove($view)) {
        die
```

```
            ("Remove DNS View object failed");
  }
  print "Removed DNS view correctly\n";

  $nview = $session->get(
      object  => "Infoblox::DHCP::View",
      name    => "my_network_view",
  );
  unless($session->remove($nview)) {
          die("Remove Network View object failed");
  }
  print "Removed network view correctly\n";

  $new_member = $session->get(
      object  => "Infoblox::Grid::Member",
      name    => "foo.com",
  );
  unless($session->remove($new_member)) {
          die("Remove Grid Member object failed");
  }
  print "Removed grid member correctly\n";

 ####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::Network, Infoblox::DHCP::View, Infoblox::DHCP::FixedAddr, Infoblox::DHCP::Range, Infoblox::Session->get()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::IPAM::DiscoveryTask - Network discovery configuration

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->modify( )
- METHODS
  - disable_ip_scanning( )
  - disable_vmware_scanning( )
  - member( )
  - mode( )
  - merge_data( )
  - networks( )
  - ping_retries( )
  - ping_timeout( )
  - state( )
  - status( )
  - tcp_ports( )
  - tcp_scan_technique( )
  - update_metadata( )
  - vservers( )
  - v_network_view( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IPAM::DiscoveryTask - Network discovery configuration

DESCRIPTION

This class represents the condiguration of network discovery jobs. Configuration parameters in this class have control over the behavior of network discovery jobs.

CONSTRUCTOR

The Infoblox::IPAM::DiscoveryTask object is automatically generated upon the successful installation of the Infoblox appliance. You do not need to create it.

SESSION METHODS

This section describes all the methods in the Infoblox::Session module that you can apply to a discovery task object.

Infoblox::Session->get( )

Use this method to retrieve matching objects from the Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get a specific IPAM network discovery task object:
 object - Required. It should be set to "Infoblox::IPAM::DiscoveryTask".
```

Example

```
my @retrieved_objs = $session->get(
    object => "Infoblox::IPAM::DiscoveryTask"
    );
```

Infoblox::Session->modify( )

Use this method to modify objects in the Infoblox appliance. See Infoblox::Session->modify() for parameters and return values.

Example

```
#Use this method to modify the ping_timeout.
$discovery_task->ping_timeout("500");
#Submit modification
my $response = $session->modify( $discovery_task);
```

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of the network discovery task.

disable_ip_scanning( )

Use this method to enable or disable IP discovery. This is a boolean value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the disable_ip_scanning attribute to "false" to enable IP discovery. Set the attribute to "true" to disable IP discovery. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable_ip_scanning attribute
my $disable_ip_scanning = $discovery_task->disable_ip_scanning();
#Modify disable_ip_scanning attribute
$discovery_task->disable_ip_scanning("true");
```

disable_vmware_scanning( )

Use this method to enable or disable VM discovery. This is a boolean value.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Set the disable_vmware_scanning attribute to "false" to enable VM discovery. Set the attribute to "true" to disable VM discovery. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable_vmware_scanning attribute
my $disable_vmware_scanning = $discovery_task->disable_vmware_scanning();
#Modify disable_vmware_scanning attribute
$discovery_task->disable_vmware_scanning("true");
```

member( )

Use this method to set or retrieve the grid member that will run the discovery. Member is required for network discovery task object.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid value is a defined Infoblox::DHCP::Member object. The default value is empty.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get member
my $member = $discovery_task->member();
#Modify member
$discovery_task->member($member1); #$member1 is an Infoblox::DHCP::Member object
```

mode( )

Use this method to set or retrieve the network discovery mode.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Network discovery mode. Valid values when setting the mode are "FULL", "ICMP", "NETBIOS" and "TCP". Default value is "FULL".

When retrieving the network discovery mode, you can get any value listed above plus the value "CSV" if a CSV discovery job is running.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get mode
my $mode = $discovery_task->mode();
#Modify mode
$discovery_task->mode("ICMP");
```

merge_data( )

Use this method to set or retrieve the merge data flag that a network discovery uses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The flag indicates whether the GM sorter replaces or merges the discovered data with existing data. The valid values are "true" and "false". The default value is "true".

Returns

If you specified a parameter, the method returns "true" when the modification succeeds, and it returns "false" when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get merge_data
my $merge_data = $discovery_task->merge_data();
#Modify merge_data
$discovery_task->merge_data("false");
```

networks( )

Use this method to retrieve the DHCP networks on which network discovery jobs search for IP addresses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::DHCP::Network objects. Networks must contain at least one Infoblox::DHCP::Network object.

Note: The discovery task object returned from the appliance may or may not contain Infoblox::DHCP::Network objects. Networks must contain at least one Infoblox::DHCP::Network object when calling modify function.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
my $network1 = Infoblox::DHCP::Network->new(
                                        "network" => "10.0.0.0/24",
                                  );
#Get networks
my $networks = $discovery_task->networks();
#Modify networks
$discovery_task->networks([$network1]);
```

ping_retries( )

Use this method to set or retrieve the network discovery ping retry times.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Network discovery ping retry times. Minimium value is 1 and maximum value is 5. Default value is 2.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ping_retries
my $ping_retries = $discovery_task->ping_retries();
#Modify ping_retries
$discovery_task->ping_retries("4");
```

ping_timeout( )

Use this method to set or retrieve the network discovery ping timeout.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Network discovery ping timeout. Minimium value is 1 ms and maximum value is 4000 ms. Default value is 1000 ms.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get ping_timeout
my $ping_timeout = $discovery_task->ping_timeout();
#Modify ping_timeout
$discovery_task->ping_timeout("500");
```

state( )

Use this method to retrieve the network discovery state. The discovery state is read-only and cannot be set.

Parameter

None

Returns

Return the state of network discovery.

Example

```
#Get state
my $state = $discovery_task->state();
```

status( )

Use this method to retrieve the network discovery status. The discovery status is read-only and cannot be set.

Parameter

None

Returns

Return the status of network discovery.

Example

```
#Get status
my $status = $discovery_task->status();
```

tcp_ports( )

Use this method to set or retrieve the TCP ports that the network discovery jobs use.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains Infoblox::IPAM::TCPPort objects. Default TCP ports are:

```
ftp               21/tcp     # File Transfer [Control]
ssh               22/tcp     # Secure Shell Login
telnet            23/tcp     #
smtp              25/tcp     # Simple Mail Transfer
finger            79/tcp     #
```

```
http              80/tcp     # World Wide Web HTTP
kerberos-sec       88/tcp     # Kerberos (v5)
pop3             110/tcp    # PostOffice V.3
rpcbind          111/tcp    # portmapper, rpcbind
auth             113/tcp    # ident, tap, Authentication Service
msrpc            135/tcp    # Microsoft RPC services
netbios-ns        137/tcp    # NETBIOS Name Service
netbios-dgm       138/tcp    # NETBIOS Datagram Service
netbios-ssn       139/tcp    # NETBIOS Session Service
imap             143/tcp    # Interim Mail Access Protocol v2
https            443/tcp    # secure http (SSL)
microsoft-ds      445/tcp    # SMB directly over IP
login            513/tcp    # BSD rlogind(8)
shell            514/tcp    # BSD rshd(8)
printer          515/tcp    # spooler (lpd)
klogin           543/tcp    # Kerberos (v4/v5)
kerberos         750/tcp    # kdc Kerberos (v4)
kdm              1024/tcp   # K Display Manager (KDE version of xdm)
NFS-or-IIS        1025/tcp   # IIS, NFS, or listener RFS remote_file_sharing
```

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get tcp_ports
my $ports = $discovery_task->tcp_ports();

#Modify tcp_ports
my $port = Infoblox::IPAM::TCPPort->new(
       number => '333',
       comment => 'test_port');
push @$ports , $port;

$discovery_task->tcp_ports($ports);
```

tcp_scan_technique( )

Use this method to set or retrieve the TCP scan technique that the network discovery jobs use.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Network discovery TCP scan technique. The scan techniques are identical to Nmap techniques. Valid values are "SYN" and "CONNECT". The default value is "SYN".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get tcp_scan_technique
my $tcp_scan_technique= $discovery_task->tcp_scan_technique();
#Modify tcp_scan_technique
$discovery_task->tcp_scan_technique("CONNECT");
```

update_metadata( )

Use this method to set or retrieve the updated metadata flag that the network discovery jobs use.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The flag indicates whether the appliance updates the metadata of the discovered objects. The valid values are "true" and "false". The default value is "true". If the value is "false", only the unmanaged objects are updated.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get update_metadata
my $update_metadata = $discovery_task->update_metadata();
#Modify update_metadata
$discovery_task->update_metadata("false");
```

vservers( )

Use this method to set or retrieve the list of VMware vSphere servers for VM discovery.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The valid value is an array reference that contains the Infoblox::IPAM::DiscoveryTask::VServer objects. The default value is a reference to an empty list.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get vservers
my $vservers = $discovery_task->vservers();
#Modify vservers
my $vserver1 = Infoblox::IPAM::DiscoveryTask::VServer->new(
       fqdn_or_ip => "vcenter.com",
       username => "test1",
       password => "test1",
   );
```

```
my $vserver2 = Infoblox::IPAM::DiscoveryTask::VServer->new(
       fqdn_or_ip => "10.2.3.4",
       username => "test2",
       password => "test2",
   );
$discovery_task->vservers([$vserver1,$vserver2]);
```

v_network_view( )

Use this method to set or retrieve the network view of a VM discovery.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

The default value is the default network view.

Parameter

The valid value is a defined Infoblox::DHCP::View object. The default value is the system-defined network view.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get network view
my $nview = $discovery_task->v_network_view();
#Modify network view, with an Infoblox::DHCP::View object
$discovery_task->v_network_view($nview);
```

SAMPLE CODE

The following sample code demonstrates different operations that can be applied to a network discovery task object, such as getting and modifying an object. This sample code also includes error handling for the operations.

#Preparation prior to the get and modify operations on a network discovery task object

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#refers to Infoblox Appliance IP address
my $host_ip   = "192.168.1.2";
my $host_name = "infoblox.localdomain";

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => $host_ip,
    username => "admin",
    password => "infoblox"
    )
    or die("Constructor for session failed: ",
               Infoblox::status_code(). ":" .Infoblox::status_detail());

print"Session created successfully.\n";

my  @result_array = $session->get(
      object  => "Infoblox::IPAM::DiscoveryTask" ,
);

my $discovery_task = $result_array[0];
unless (ref($discovery_task) eq 'Infoblox::IPAM::DiscoveryTask') {
       die("Get discovery task failed");
}
print "Get discovery task successfully\n";

#Create the Network object
my $network = Infoblox::DHCP::Network->new(
       network => "10.0.0.0/24",
);

$session->add($network)
   or die("Add network failed: " .
               $session->status_code() . ":" . $session->status_detail());
#modify networks
$discovery_task->networks([$network]);

# modify ping_timeout
$discovery_task->ping_timeout("500");

# modify tcp_ports
my $ports = $discovery_task->tcp_ports();

my $port = Infoblox::IPAM::TCPPort->new(
       number => '333',
       comment => 'test_port');
push @$ports , $port;

$discovery_task->tcp_ports($ports);

# submit changes
$session->modify( $discovery_task )
       or die("Modify discovery task failed: " .
               $session->status_code() . ":" . $session->status_detail());
print "Modify discovery task successfully\n";

# clean up
# remove network
$session->remove( $network)
   or die("Remove network failed: " .
          $session->status_code() . ":" . $session->status_detail());

####### Program Ends ###########################
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IPAM::DiscoveryTask::VServer, **Infoblox::DHCP::Member**, **Infoblox::DHCP::Network**, **Infoblox::IPAM::TCPPort**, **Infoblox::Session**, **Infoblox::Session->get()**, **Infoblox::Session->modify()**

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::IPAM::DiscoveryTask::VServer - VMWare discovery server object.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- MODULE METHODS
  - Infoblox::IPAM::DiscoveryTask->vservers( )
- METHODS
  - disable( )
  - encryption( )
  - fqdn_or_ip( )
  - password( )
  - port( )
  - username( )
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IPAM::DiscoveryTask::VServer - VMWare discovery server object.

DESCRIPTION

This object contains information about the vSphere servers on which the VM discovery is performed.

CONSTRUCTOR

```
my $vserver = Infoblox::IPAM::DiscoveryTask::VServer->new(
        fqdn_or_ip   => $string,           #Required.
        password     => $string,           #Required.
        username     => $string,           #Required.
        disable      => "true" | "false",  #Optional. Default is "false".
        encryption   => "true" | "false",  #Optional. Default is "true".
        port         => $num,              #Optional. Default is 443.
);
```

MODULE METHODS

You can apply the following functions to a VMware vSphere server.

Infoblox::IPAM::DiscoveryTask->vservers( )

Use this function to specify the vSphere server on which the VM discovery is performed. See Infoblox::IPAM::DiscoveryTask->vservers( ) for parameters and returned values.

Example

```
my $vserver1 = Infoblox::IPAM::DiscoveryTask::VServer->new(
       fqdn_or_ip => "vcenter.com",
       username => "test1",
       password => "test1",
   );
my $vserver2 = Infoblox::IPAM::DiscoveryTask::VServer->new(
       fqdn_or_ip => "10.2.3.4",
       username => "test2",
       password => "test2",
   );
 #Modifying vserver
 $discovery_task->vservers([$vserver1, $vserver2]);
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of a vSphere server.

disable( )

Use this method to set or retrieve the disable flag of a vSphere server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to disable the vSphere server or "false" to enable the server. The default value is "false".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get disable flag
my $disable = $vserver->disable();
#Modify disable
$vserver->disable("true");
```

encryption( )

Use this method to set or retrieve the encryption flag of a vSphere server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Specify "true" to enable encryption or specify "false" to disable encryption on the vSphere server. The default value is "true".

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get encryption
my $encryption = $vserver->encryption();
#Modify encryption
$vserver->encryption("false");
```

fqdn_or_ip( )

Use this method to retrieve the FQDN (Fully Qualified Domain Name) or IP address of the vShpere server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The string that contains the FQDN or the IP address of the vSphere server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get fqdn_or_ip
my $fqdn_or_ip = $vserver->fqdn_or_ip();
#Modify fqdn_or_ip
$vserver->fqdn_or_ip("vdiscovery.example.com");
```

password( )

Use this method to retrieve the user password on the vSphere server.

Parameter

String containing password.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the ssha encrypted attribute value.

Example

```
#Get password
my $password = $vserver->password();
#Modify password
$vserver->password("infobloxx");
```

port( )

Use this method to retrieve the connection port that the vSphere server uses.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Valid values are between 1 and 65535. The default value is 443.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get port
my $port = $vserver->port();
#Modify port
$vserver->port(444);
```

username( )

Use this method to set or retrieve the user name on the VSphere server.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

The username in the string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
#Get username
my $username = $vserver->username();
#Modify username
$vserver->username("someuser");
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IPAM::DiscoveryTask, Infoblox::IPAM::DiscoveryTask->vservers()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::IPAM::Statistics - Manages IPAM statistics.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
  - Infoblox::Session->get( )
  - Infoblox::Session->search( )
- METHODS
  - utilization( )
  - conflict_count( )
  - unmanaged_count( )
  - utilization_update( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IPAM::Statistics - Manages IPAM statistics.

DESCRIPTION

The IPAM statistics object is used to view the IPAM statistics of the network or network container in an Infoblox appliance.

CONSTRUCTOR

The Infoblox::IPAM::Statistics object does not require manual construction.

SESSION METHODS

This section describes all the methods in an Infoblox::Session that can be applied to a IPAM statistics object.

Infoblox::Session->get( )

Use this method to retrieve the existing objects from an Infoblox appliance. See Infoblox::Session->get() for parameters and return values.

Key Reference

```
Apply the following attributes to get the IPAM statistics:

network      - Required. An IPv4 address in dotted decimal notation followed by the network prefix
(e.g 1.0.0.0/8).
network_view - Optional. The name of the network view in which the network object is located. By
default, all network views are searched.
```

Example

```
my $ipam_stats = $session->get(
        object  => "Infoblox::IPAM::Statistics",
        network => "1.0.0.0/8",
        network_view => "my_network_view");
```

Infoblox::Session->search( )

Use this method to search for an IPAM statistics object from an Infoblox appliance. See Infoblox::Session->search() for parameters and return values.

Key Reference

```
Apply the following attributes to get the IPAM statistics:

network      - Required. An IPv4 address in dotted decimal notation followed by the network prefix
(e.g 1.0.0.0/8) (regular expression).
network_view - Optional. The name of the network view in which the network object is located. By
default, all network views are searched.
```

Example

```
my $ipam_stats = $session->search(
        object  => "Infoblox::IPAM::Statistics",
        network => '1\.0\.0\..*',
        network_view => "my_network_view");
```

METHODS

This section describes all the methods that can be used to retrieve the attribute values of a IPAM statistics object.

utilization( )

Use this method to retrieve the network utilization.

Parameter

none

Returns

The method returns the network utilization in percentage.

Example

```
#Get utilization
my $utilization = $ipam_stats->utilization();
```

conflict_count( )

Use this method to retrieve the number of conflicts discovered via network discovery. This method is only valid for a Network object.

Parameter

none

Returns

The method returns the number of discovered conflicts. For a NetworkContainer object, the return value is undefined.

Example

```
#Get conflict_count
my $conflict_count = $ipam_stats->conflict_count();
```

unmanaged_count( )

Use this method to retrieve the number of unmanaged IP addresses as discovered by network discovery. This method is only valid for a Network object.

Parameter

none

Returns

The method returns the number of unmanaged IP addresses. For a NetworkContainer object, the return value is undefined.

Example

```
#Get unmanaged_count
my $unmanaged_count = $ipam_stats->unmanaged_count();
```

utilization_update( )

Parameter

none

Returns

The method returns a timestamp value in the ISO 8601 extended format for date and time (e.g. 2008-10-21T10:56:33Z or 2008-10-21T10:56:33.001Z). For a NetworkContainer object, the return value is undefined.

Example

```
#Get utilization_update
my $utilization_update = $ipam_stats->utilization_update();
```

SAMPLE CODE

The following sample code demonstrates the get and search session methods on an IPAM statistics object. Also, this sample code also includes error handling for the operations.

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;
```

#Preparation prior to getting IPAM statistics

```
#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
unless ($session) {
   die("Construct session failed: ",
       Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Session created successfully\n";

my $member = Infoblox::DHCP::Member->new(
    name     => "infoblox.localdomain",
    ipv4addr => "192.168.1.2",
);

unless($member) {
     die("Construct member failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Member object created successfully\n";

#Create a Network View object
my $network_view = Infoblox::DHCP::View->new(
                   name => "my_network_view",
);
unless($network_view) {
            die("Construct Network View object failed: ",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network View object created successfully\n";

#Add the Network View object into the Infoblox appliance through a session
$session->add($network_view)
     or die("Add Network View object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Network View added successfully\n";

#Create the Network object with this member
my $network = Infoblox::DHCP::Network->new(
                   network => "1.0.0.0/255.0.0.0",
                   members => [ $member ],
                   network_view => $network_view,
);
unless($network) {
            die("Construct Network object failed: ",
                  Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "Network object created successfully\n";

#Add the Network object into the Infoblox appliance through a session
$session->add($network)
     or die("Add Network object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "Network added successfully\n";

#Create the DHCP Range object with this member
my $dhcp_range = Infoblox::DHCP::Range->new(
    network    => "1.0.0.0/8",
    start_addr => "1.0.0.1",
    end_addr   => "1.0.0.10",
    member     => $member,
    network_view => $network_view,
);
```

```
unless($dhcp_range) {
     die("Construct DHCP Range failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP range object created successfully\n";

$session->add($dhcp_range)
     or die("Add DHCP range object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DHCP range added successfully\n";

#Create the Fixed Address object
my $fixed_address = Infoblox::DHCP::FixedAddr->new(
                                                "network" => "1.0.0.0/8",
                                                "ipv4addr" => "1.0.0.2",
                                                "mac" => "22:22:22:22:22:22",
                                                "network_view" => $network_view,
                                                );
unless($fixed_address) {
     die("Construct DHCP Fixed address failed: ",
           Infoblox::status_code() . ":" . Infoblox::status_detail());
}
print "DHCP Fixed address object created successfully\n";

$session->add($fixed_address)
     or die("Add DHCP Fixed address object failed: ",
           $session->status_code() . ":" . $session->status_detail());
print "DHCP Fixed address added successfully\n";
```

#Get the IPAM statistics

```
my @result_array = $session->get(
         object  => "Infoblox::IPAM::Statistics",
         network => "1.0.0.0/8",
         network_view => "my_network_view");

my $ipam_stats = $result_array[0];
unless ($ipam_stats) {
     die("Get IPAM statistics failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print "Get IPAM Statistics successful\n";

print "Utilization for '1.0.0.0/8': ", $ipam_stats->utilization(), "\n";
```

#Search for the IPAM statistics

```
my @result_array = $session->search(
         object  => "Infoblox::IPAM::Statistics",
         network => '1\.0\.0\..*',
         network_view => "my_network_view");

my $ipam_stats = $result_array[0];
unless ($ipam_stats) {
     die("Get IPAM statistics failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
print "Get IPAM Statistics successful\n";

print "Utilization for '1.0.0.0/8': ", $ipam_stats->utilization(), "\n";
```

#Clean up

```
# Remove the IPAM Statistics object that was just retrieved
$session->remove($ipam_stats);

# Retrieve and remove the Network View object
my $network_view = $session->get(
        object => "Infoblox::DHCP::View",
        name => "my_network_view",
);
unless ($network_view) {
     die("Get Network View failed: ",
           $session->status_code() . ":" . $session->status_detail());
}
$session->remove($network_view);
```

```
####PROGRAM ENDS####
```

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::Session, Infoblox::DHCP::Network, Infoblox::DHCP::View, Infoblox::DHCP::FixedAddr, Infoblox::DHCP::Range, Infoblox::Session->add(), Infoblox::Session->get()

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::IPAM::TCPPort - Network discovery TCP port

- NAME
- DESCRIPTION
- CONSTRUCTOR
- SESSION METHODS
- METHODS
  - comment( )
  - number( )
- SAMPLE CODE
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::IPAM::TCPPort - Network discovery TCP port

DESCRIPTION

This class represents the TCP port that network discovery jobs use to find IP addresses.

CONSTRUCTOR

```
my $tcp_port = Infoblox::IPAM::TCPPort->new(
     number            => $string,             #Required.
     comment           => $string,             #Optional.
);
```

SESSION METHODS

This object does not have a session method. You must use it with Infoblox::IPAM::DiscoveryTask class.

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a TCP port.

comment( )

Use this method to set or retrieve a descriptive comment for a TCP port.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a descriptive comment for the TCP port in string format.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get comment
my $comment = $tcp_port->comment();
# Modify comment
$tcp_port->comment("new port");
```

number( )

Use this method to set or retrieve the TCP port number.

Include the specified parameter to set the attribute value. Omit the parameter to retrieve the attribute value.

Parameter

Enter a number the TCP port.

Returns

If you specified a parameter, the method returns true when the modification succeeds, and returns false when the operation fails.

If you did not specify a parameter, the method returns the attribute value.

Example

```
# Get port number
my $port_number = $tcp_port->number();
# Modify port number
$tcp_port->number("222");
```

SAMPLE CODE

Please see sample code in Infoblox::IPAM::DiscoveryTask.

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::IPAM::DiscoveryTask

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Infoblox::Session - Session object to manipulate data within a grid.

- NAME
- DESCRIPTION
- CONSTRUCTOR
- METHODS
  - add( )
  - clear_dns_cache( )
  - clear_unmanaged( )
  - copy_records( )
  - empty_recycle_bin( )
  - export_data( )
  - fill_partial_object( )
  - gen_tsig_key( )
  - get( )
  - import_data( )
  - join_grid( )
  - modify( )
  - new_cursor( )
  - network_discovery_control( )
  - remove( )
  - remove_data( )
  - remove_eap_ca_cert( )
  - remove_tftp_file( )
  - reset_discovery_timestamp( )
  - reset_discovery_data( )
  - restart( )
  - restart_status( )
  - search( )
  - server_version( )
  - set_dhcp_failover_partner_down( )
  - set_dhcp_failover_secondary_recover( )
  - status_code( )
  - status_detail( )
- SAMPLE CODE
- SCHEDULED OBJECT USAGE DETAILS
- AUTHOR
- SEE ALSO
- COPYRIGHT

NAME

Infoblox::Session - Session object to manipulate data within a grid.

DESCRIPTION

The Infoblox::Session object is the key object that is used to manipulate data within a grid. Creating a Session object establishes a communications channel with the grid master.

CONSTRUCTOR

```
my $session = Infoblox::Session->new(
    master   => $ipv4addr, #Required
    password => $string,   #Required
    username => $string,   #Required
    timeout  => $num,      #Optional, default is to use the LWP::UserAgent default timeout (180 seconds)
    connection_timeout => $num  #Optional, see below for more details
);
```

If connection_timeout is provided and is greater than zero, the API will try to connect to the master's IP address with the specified connection_timeout first; and if it's successful, it will proceed to the log-in process.

METHODS

This section describes all the methods that you can use to configure and retrieve the attribute values of a Session object.

add( )

Use this method to add an object to the Infoblox appliance.

Parameter

The Infoblox object to be created in the Infoblox appliance (e.g.,Infoblox::DNS::Record::A). See individual objects for the necessary parameters.

scheduled_at - Optional. Scheduled date and time in ISO 8601 extended format (e.g., 2008-10-21T10:56:00Z or 2008-10-21T10:56:00.001Z). See details.

Returns

The method returns true when the operation succeeds, and returns false when the operation fails. When an operation fails, the error code and error message are reported in the `status_code()` and `status_detail()` attributes.

Example

```
#Construct an object
my $bind_a = Infoblox::DNS::Record::A->new (
    ipv4addr => "1.1.1.1",
    name     => "bind_a.domain.com" );
# Submit for add
my $response = $session->add( $bind_a );
```

clear_dns_cache( )

Use this method to clear the DNS cache.

Parameter

member - Required. The IPv4 address of the member on which you want to clear the DNS cache.

domain - Optional. The name of the domain to which the clear operation will be limited. The default is all domains.

view - Optional. The name of the DNS view to which the clear operation will be limited. The default is all views.

Returns

The method returns true when the operation succeeds, and returns false when the operation fails. When an operation fails, the error code and error message are reported in the `status_code()` and `status_detail()` attributes.

Example

```
# clear the entire DNS cache
my $response = $session->clear_dns_cache(
       member => "192.168.1.2");

unless($response) {
   die("Clear DNS cache failed");
}

# clear a DNS view from the DNS cache
$response = $session->clear_dns_cache(
       member => "192.168.1.2",
       view => "new_view",
);

unless($response) {
   die("Clear DNS view from DNS cache failed");
}

# clear a domain from the DNS cache
$response = $session->clear_dns_cache(
       member => "192.168.1.2",
       domain => "foo.bar.com",
);

unless($response) {
   die("Clear the domain \"foo.bar.com\" from DNS cache failed");
}
```

clear_unmanaged( )

Use this method to clear unmanaged addresses.

Parameter

type - Required. Valid values are "GLOBAL", "NETWORK", and "IP_ADDRESS".

network - Optional. Required when type is "NETWORK" or "IP_ADDRESS".

network_view - Optional. The network view to which the clear operation will be limited. The default value is the system-defined default network view.

ip_address - Optional. Required when type is "IP_ADDRESS".

Returns

The method returns true when the operation succeeds, and returns false when the operation fails. When an operation fails, the error code and error message are reported in the `status_code()` and `status_detail()` attributes.

Example

```
# type is GLOBAL
my $response = $session->clear_unmanaged(
       type => 'GLOBAL');

unless($response) {
   die("Clear unmanaged address failed");
}

# with a non-default network view
my $nview = Infoblox::DHCP::View->new(
       name => 'netview',
);
my $response = $session->clear_unmanaged(
       type => 'GLOBAL',
       network_view => $nview);

unless($response) {
   die("Clear unmanaged address failed");
}

# type is NETWORK
$response = $session->clear_unmanaged(
       type => 'NETWORK',
       network => '10.0.0.0/24');

unless($response) {
   die("Clear unmanaged address failed");
}

# type is IP_ADDRESS
$response = $session->clear_unmanaged(
       type => 'IP_ADDRESS',
       network => '10.0.0.0/24',
       ip_address => '10.0.0.10');

unless($response) {
   die("Clear unmanaged address failed");
}

####PROGRAM ENDS####
```

copy_records( )

Copy records between DNS views.

Parameter

dest_view - Required. The destination DNS view.

dest_zone - Required. The destination DNS zone.

view - Required. The source DNS view.

zone - Required. The source DNS zone.

clear_dest_zone - Optional. Remove all records in the destination zone before copying the records. Default is "false".

copy_all_rr - Optional. Copy all records. Default is "false".

copy_bind_srv - Optional. Copy SRV records. Default is "false".

copy_bind_txt - Optional. Copy TXT records. Default is "false".

copy_bind_ptr - Optional. Copy PTR records. Default is "false".

copy_bulk_host - Optional. Copy Bulk Host records. Default is "false".

copy_bind_a - Optional. Copy A records. Default is "false".

copy_bind_cname - Optional. Copy CNAME records. Default is "false".

copy_bind_dname - Optional. Copy DNAME records. Default is "false".

copy_bind_mx - Optional. Copy MX records. Default is "false".

copy_host - Optional. Copy Host records. Default is "false".

replace_existing_records - Optional. Replace any existing records with the copied records if duplicate records are not allowed. Default is "false".

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#copy records from default view to new_view
$session->copy_records(
    view                    => "default",
    zone                    => "domain.com",
    dest_view               => "new_view",
    dest_zone               => "domain.com",
    copy_bind_srv           => "true",
    copy_bind_txt           => "true",
    copy_bind_ptr           => "false",
    copy_bulk_host          => "false",
    copy_bind_a             => "true",
    copy_bind_cname         => "true",
    copy_bind_dname         => "true",
    copy_bind_mx            => "true",
    copy_host               => "true",
    replace_existing_records => "false",
    clear_dest_zone         => "false",
    copy_all_rr             => "false" );
```

empty_recycle_bin( )

Remove the DNS/DHCP data from the recycle bin.

Parameter

none

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#empty recycle bin
my $response = $session->empty_recycle_bin ();
```

export_data( )

Use this method to download data from the Infoblox appliance to the local system.

Parameter

path - Required. The absolute path of the destination file for the data you want to export.

type - Required. The data type to export. Following is a list of available data types for export:

```
 +-------------------------------------------------------------------------------------------
---------------------------+
 | admin_cert       - Export a member certificate. Exported file is in "pem" format.
|
 | member           - Required. The member from which you want to export the certificate.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | backup         - Export Infoblox appliance database. Exported file is in "tar.gz" format.
|
 | backup_type    - Optional. Valid values are LOCAL and SCP. Default is LOCAL.
|
 | backup_server  - Optional. The IPv4 address of the backup server.
```

```
|
| user           - Optional. The user name of the backup server.
|
| password       - Optional. The user password of the backup server.
|
+-----------------------------------------------------------------------------------------------------------------+
| captive_portal_cert   - Exports the server certificate of the captive portal.
|
| dn                    - Required. The distinguished name (dn) of the certificate subject.
|
+-----------------------------------------------------------------------------------------------------------------+
| csv               - Export data in a CSV file
|
|
|
| object_type       - Required. The object type of the object to be exported
|
| separator         - Optional. CSV separator character. Valid values are 'comma', 'semicolon', 'space' and 'tab'.  |
|                             The default is 'comma'.
|
|
|
| <various>         - Optional. A list of search parameters that identify the object(s) you want to
|                             export. See the individual API object pages for the parameters, which are           |
|                             the same as the parameters that are passed to $session->search(). This document     |
|                             provides information about the supported objects.
|
|
|
|                             If the objects you want to export have a search parameter named type, path,         |
|                             object_type or separator, pass the search parameter with a prepended
|
|                             underscore, such as '_type', '_separator', etc.
|
+-----------------------------------------------------------------------------------------------------------------+
| csv_error_log  - Export the CSV import error log file
|
| import_id      - Optional. The ID of the import task for the log file you want to export. If not specified,       |
|                  the current export task ID is used.
|
+-----------------------------------------------------------------------------------------------------------------+
| dnsCache       - Export the DNS dnsCache.db file of a member. Exported file is in "tar.gz" format. The            |
|                  following parameter is also available:
|
| member         - Required. The member from which you want to download the dns cache file.
|
+-----------------------------------------------------------------------------------------------------------------+
| dhcpd_conf     - Export the DHCP configuration file of a member. Exported file is in "tar.gz" format. The        |
|                  following parameter is also available:
|
| member         - Required. The member from which you want to download the DHCP configuration file.
|
+-----------------------------------------------------------------------------------------------------------------+
| dhcpdv6_conf   - Export the DHCPv6 configuration file of a member. Exported file is in "tar.gz" format. The       |
|                  following parameter is also available:
|
| member         - Required. The member from which you want to download the DHCPv6 configuration file.              |
+-----------------------------------------------------------------------------------------------------------------+
| dhcp_expert_mode_config - Export the DHCP configuration file of a member in expert mode. Exported file is in      |
|                           "dhcpd.conf" format. The following parameter is also available:
|
| member                  - Required. The member from which you want to download the DHCP configuration export mode |
|                           file.
|
+-----------------------------------------------------------------------------------------------------------------+
| dns_conf     - Export the DNS configuration file of a member. Exported file is in "tar.gz" format. The            |
|                following parameter is also available:
|
| member       - Required. The member from which you want to download the DHCP configuration file.
|
+-----------------------------------------------------------------------------------------------------------------+
| ds_records     - Export the keyset associated with a given zone. This is useful for sending to the parent zone    |
|                  for signing.
|
```

```
 | zone           - Required. The zone that contain the keys to be exported.
|
 |                  Specify either the FQDN of the zone or the Infoblox::DNS::Zone object.
|
 | view           - Optional. The DNS view that contains the zone. Specify either the DNS view name
|
 |                  or Infoblox::DNS::View object. The default is the "default" view.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | eap_ca_cert    - Export an Extensible Authentication Protocol Certificate Authority (EAP CA)
certificate from the |
 |                  Infoblox appliance. Exported file is in "pem" format. The following parameter is
also            |
 |                  available:
|
 | dn             - Required. The distinguished name (dn) of your certificate's subject.
|
 |                           Examples of dn: CN="Testing",OU="Sales",O="Infoblox",L="Santa
Clara",ST="CA",C="US"    |
 |                                           O="My Company",ST="CA",C="US"
|
 |                                           CN="Test Common Name"
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | generate_captive_portal_csr    - Generate and export a certificate signing request (CSR) for a
captive portal     |
 |                    Exported file is in "pem" format.
|
 | cn               - Required. Common name for the certificate.
|
 | member           - Required. The member for which you want to generate the CSR.
|
 | key_size         - Optional.  Number of bits in the key. It must be either 1024 or 2048. Default
is 2048.         |
 | org_unit         - Optional.  Organizational unit used to construct the distinguished name of the
subject.         |
 | org              - Optional.  Organization used to construct the distinguished name of the
subject.            |
 | locality         - Optional.  Location (e.g., city, town) used to construct the distinguished name
of the subject.|
 | state            - Optional.  State or province used to construct the distinguished name of the
subject.          |
 | country          - Optional.  2-letter country code used to construct the distinguished name of
the subject.      |
 | email            - Optional.  Contact email address used to construct the distinguished name of
the subject.      |
 | comment          - Optional.  A descriptive comment about the certificate to be signed.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | generate_csr     - Generate and export a certificate signing request (CSR) for a grid member
|
 |                    Exported file is in "pem" format.
|
 | cn               - Required. Common name for the certificate.
|
 | member           - Required. The member for which you want to generate the CSR.
|
 | key_size         - Optional.  Number of bits in the key. It must be either 1024 or 2048. Default
is 2048.         |
 | org_unit         - Optional.  Organizational unit used to construct the distinguished name of the
subject.         |
 | org              - Optional.  Organization used to construct the distinguished name of the
subject.            |
 | locality         - Optional.  Location (e.g., city, town) used to construct the distinguished name
of the subject.|
 | state            - Optional.  State or province used to construct the distinguished name of the
subject.          |
 | country          - Optional.  2-letter country code used to construct the distinguished name of
the subject.      |
 | email            - Optional.  Contact e-mail address used to construct the distinguished name of
the subject.     |
 | comment          - Optional.  A descriptive comment about the certificate to be signed.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | generate_ifmap_csr - Generate and export a certificate signing request (CSR) for the IF-MAP
server                |
 |                    Exported file is in "pem" format.
|
 | cn               - Required. Common name for the certificate.
|
 | member           - Required. The member for which you want to generate the CSR.
|
 | key_size         - Optional.  Number of bits in the key. It must be either 1024 or 2048. Default
is 2048.         |
 | org_unit         - Optional.  Organizational unit used to construct the distinguished name of the
subject.         |
 | org              - Optional.  Organization used to construct the distinguished name of the
subject.            |
 | locality         - Optional.  Location (e.g., city, town) used to construct the distinguished name
of the subject.|
 | state            - Optional.  State or province used to construct the distinguished name of the
subject.          |
 | country          - Optional.  2-letter country code used to construct the distinguished name of
```

```
the subject.      |
 | email           - Optional.  Contact email address used to construct the distinguished name of
the subject.      |
 | comment         - Optional.  A descriptive comment about the certificate to be signed.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | generate_ifmap_federation_csr - Generate and export a certificate signing request (CSR) for IF-
MAP Federation     |
 |                   Exported file is in "pem" format.
|
 | cn              - Required. Common name for the certificate.
|
 | member          - Required. The member for which you want to generate the CSR.
|
 | key_size        - Optional.  Number of bits in the key. It must be either 1024 or 2048. Default
is 2048.         |
 | org_unit        - Optional.  Organizational unit used to construct the distinguished name of the
subject.        |
 | org             - Optional.  Organization used to construct the distinguished name of the
subject.          |
 | locality        - Optional.  Location (e.g., city, town) used to construct the distinguished name
of the subject.|
 | state           - Optional.  State or province used to construct the distinguished name of the
subject.          |
 | country         - Optional.  2-letter country code used to construct the distinguished name of
the subject.      |
 | email           - Optional.  Contact email address used to construct the distinguished name of
the subject.     |
 | comment         - Optional.  A descriptive comment about the certificate to be signed.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | generate_self_signed_admin_cert - Generate and export a self-signed certificate for a grid member
|
 |                   Exported file is in "pem" format.
|
 | cn              - Required. Common name for the certificate.
|
 | days_valid      - Required. Validity period for the certificate.
|
 | member          - Required. The member for which you want to generate the certificate.
|
 | key_size        - Optional.  Number of bits in the key. It must be either 1024 or 2048. Default
is 2048.         |
 | org_unit        - Optional.  Organizational unit used to construct the distinguished name of the
subject.        |
 | org             - Optional.  Organization used to construct the distinguished name of the
subject.          |
 | locality        - Optional.  Location (e.g., city, town) used to construct the distinguished name
of the subject.|
 | state           - Optional.  State or province used to construct the distinguished name of the
subject.          |
 | country         - Optional.  2-letter country code used to construct the distinguished name of
the subject.      |
 | email           - Optional.  Contact email address used to construct the distinguished name of
the subject.      |
 | comment         - Optional.  A descriptive comment about the certificate to be signed.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | generate_self_signed_captive_portal_cert - Generate and export a self-signed certificate for a
captive portal     |
 |                   Exported file is in "pem" format.
|
 | cn              - Required. Common name for the certificate.
|
 | days_valid      - Required. Validity period for the certificate.
|
 | member          - Required. The member for which you want to generate the certificate.
|
 | key_size        - Optional.  Number of bits in the key. It must be either 1024 or 2048. Default
is 2048.         |
 | org_unit        - Optional.  Organizational unit used to construct the distinguished name of the
subject.        |
 | org             - Optional.  Organization used to construct the distinguished name of the
subject.          |
 | locality        - Optional.  Location (e.g., city, town) used to construct the distinguished name
of the subject.|
 | state           - Optional.  State or province used to construct the distinguished name of the
subject.          |
 | country         - Optional.  2-letter country code used to construct the distinguished name of
the subject.      |
 | email           - Optional.  Contact email address used to construct the distinguished name of
the subject.      |
 | comment         - Optional.  A descriptive comment about the certificate to be signed.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | generate_self_signed_ifmap_cert - Generate and export a self-signed certificate for the IF-MAP
server             |
 |                   Exported file is in "pem" format.
|
 | cn              - Required. Common name for the certificate.
|
 | days_valid      - Required. Validity period for the certificate.
|
```

```
 | member           - Required. The member for which you want to generate the certificate.
|
 | key_size         - Optional.  Number of bits in the key. It must be either 1024 or 2048. Default
is 2048.        |
 | org_unit         - Optional.  Organizational unit used to construct the distinguished name of the
subject.        |
 | org              - Optional.  Organization used to construct the distinguished name of the
subject.          |
 | locality         - Optional.  Location (e.g., city, town) used to construct the distinguished name
of the subject.|
 | state            - Optional.  State or province used to construct the distinguished name of the
subject.          |
 | country          - Optional.  2-letter country code used to construct the distinguished name of
the subject.      |
 | email            - Optional.  Contact email address used to construct the distinguished name of
the subject.      |
 | comment          - Optional.  A descriptive comment about the certificate to be signed.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | generate_self_signed_ifmap_federation_cert - Generate and export a self-signed certificate for
IF-MAP Federation  |
 |                    Exported file is in "pem" format.
|
 | cn               - Required. Common name for the certificate.
|
 | days_valid       - Required. Validity period for the certificate.
|
 | member           - Required. The member for which you want to generate the certificate.
|
 | key_size         - Optional.  Number of bits in the key. It must be either 1024 or 2048. Default
is 2048.        |
 | org_unit         - Optional.  Organizational unit used to construct the distinguished name of the
subject.        |
 | org              - Optional.  Organization used to construct the distinguished name of the
subject.          |
 | locality         - Optional.  Location (e.g., city, town) used to construct the distinguished name
of the subject.|
 | state            - Optional.  State or province used to construct the distinguished name of the
subject.          |
 | country          - Optional.  2-letter country code used to construct the distinguished name of
the subject.      |
 | email            - Optional.  Contact e-mail address used to construct the distinguished name of
the subject.     |
 | comment          - Optional.  A descriptive comment about the certificate to be signed.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | ifmap_cert       - Export an IF-MAP server certificate. Exported file is in "pem" format.
|
 | member           - Required. The member from which you want to export the certificate.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | ifmap_cit_schema - Export a custom identifier type. The output file is the original imported XSD
schema.          |
 | cit              - Required. A reference to the custom identifier type (CIT) object associated
with the desired   |
 |                    schema.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | ifmap_federation_cert       - Export an IF-MAP Federation certificate. Exported file is in "pem"
format.          |
 | member                      - Required. The member from which you want to export the certificate.
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | log_files        - Export the specified log files from a member. Exported file is in "tar.gz"
format. The        |
 |                    following parameters are also available:
|
 | member           - The member from which you are downloading the logs. You must specify a member
or a Microsoft(r)|
 |                    server.
|
 | msserver         - The Microsoft server from which you are downloading the logs. You must specify
a member or     |
 |                    a Microsoft server.
|
 | node_type        - Optional. Node type. Valid values are "active" and "passive". The default is
"active".        |
 | log_type         - Optional. Log type. Valid values are "audit", "ifmap", "msmgmt", and "syslog".
The default     |
 |                    is "syslog".
|
 +-----------------------------------------------------------------------------------------------
---------------------------+
 | lease_history  - Export the DHCP lease history. Exported file is in "gz" format.
|
 | start_time     - Optional. The start time of the exported lease history in the format yyyy-mm-
ddThh:mm:ss,       |
 |                  yyyy/mm/dd hh:mm:ss, or mm/dd/yyyy hh:mm:ss.
|
 | end_time       - Optional. The end time of the exported lease history in the format yyyy-mm-
ddThh:mm:ss,         |
 |                  yyyy/mm/dd hh:mm:ss, or mm/dd/yyyy hh:mm:ss.
```

```
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | snmp_mibs      - Export the Infoblox SNMP MIB files. Exported file is in "tar.gz" format.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | support_bundle        - Export the support bundle from a member. Exported file is in "tar.gz"
format. The         |
 |                         following parameters are also available:
|
 | member                - Required. The member from which you are downloading the support bundle.
|
 | filter_optional_files - Optional. Filter optional files. Default is "true".
|
 | core_files            - Optional. Include core files in support bundle. Default is "true".
|
 | log_files             - Optional. Include log files in support bundle. Default is "true".
|
 | rotate_log_files      - Optional. Include rotate_log files in support bundle. Default is "true".
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | serial_numbers - Export the serial numbers of all nodes in the grid. Exported file is in plain
text format.       |
 +-------------------------------------------------------------------------------------------
---------------------------+
 | trust_anchors  - Export trust anchors, i.e. our public keys to allow creation of the island of
security.          |
 |
|
 | zone           - Required. The zone that contains the keys to be exported.
|
 |                  Specify either the FQDN of the zone or the Infoblox::DNS::Zone object.
|
 | view           - Optional. The DNS view that contains the zone. Specify either the DNS view name
or the           |
 |                  Infoblox::DNS::View object. The default is the "default" view.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
```

For CSV export, the following objects are supported:

```
Infoblox::DHCP::Filter::MAC
Infoblox::DHCP::Filter::MatchRule
Infoblox::DHCP::Filter::NAC
Infoblox::DHCP::Filter::Option
Infoblox::DHCP::Filter::RelayAgent
Infoblox::DHCP::FixedAddr
Infoblox::DHCP::IPv6Network
Infoblox::DHCP::IPv6DhcpRange
Infoblox::DHCP::IPv6FixedAddress
Infoblox::DHCP::IPv6SharedNetwork
Infoblox::DHCP::IPv6OptionDefinition
Infoblox::DHCP::IPv6OptionSpace
Infoblox::DHCP::MAC
Infoblox::DHCP::Network
Infoblox::DHCP::OptionDefinition
Infoblox::DHCP::OptionSpace
Infoblox::DHCP::Range
Infoblox::DHCP::SharedNetwork
Infoblox::Grid::DNS::DNS64Group
Infoblox::DNS::AllRecords
Infoblox::DNS::Host
Infoblox::DNS::Record::A
Infoblox::DNS::Record::AAAA
Infoblox::DNS::Record::CNAME
Infoblox::DNS::Record::DNAME
Infoblox::DNS::Record::MX
Infoblox::DNS::Record::NAPTR
Infoblox::DNS::Record::NS
Infoblox::DNS::Record::PTR
Infoblox::DNS::Record::SRV
Infoblox::DNS::Record::TXT
Infoblox::DNS::Ruleset
Infoblox::DNS::Zone (only authoritative zones)
```

For more information about the CSV file format please refer to the Infoblox CSV Import Reference.

Returns

The method returns true when the modification succeeds, and returns false when the operation fails. The exported data is stored in the path specified.

Example

```
#Export backup
$session->export_data(
    type   => "backup",
    path   => "/tmp/infoblox/database.tar.gz" );

#Secured backup
```

```
my $result = $session->export_data(
    type   => "backup",
    backup_type => "SCP",
    path   => "/tmp/infoblox/database.tar.gz",
    backup_server => "192,168,0,5",
    user => "testuser",
    password => "test-password",
    );

#CSV export
$session->export_data(
                      type        => "csv",
                      path        => "/tmp/infoblox/exported_data.csv",
                      object_type => "Infoblox::DNS::Record::A",
                      separator   => "comma",

                      # search parameters
                      name        => ".*test_exportimport.com");

#CSV error log export
$session->export_data(
                      type      => "csv_error_log",
                      path      => "/tmp/infoblox/error_log.txt",
                      import_id => 12345);

#Export dnsCache.db
my $result = $session->export_data(
    type   => "dnsCache",
    path   => "/tmp/infoblox/dnsCache.tar.gz",
    member => "192.168.1.2" );

#Export dhcpd_conf
my $result = $session->export_data(
    type   => "dhcpd_conf",
    path   => "/tmp/infoblox/dhcpd_conf.tar.gz",
    member => "192.168.1.2" );

#Export dhcpdv6_conf
my $result = $session->export_data(
    type   => "dhcpdv6_conf",
    path   => "/tmp/infoblox/dhcpdv6_conf.tar.gz",
    member => "192.168.1.2" );

#Export dhcp_expert_mode_config
my $result = $session->export_data(
    type   => "dhcp_expert_mode_config",
    path   => "/tmp/infoblox/dhcpd_expert.conf",
    member => "192.168.1.2" );

#Export dns_conf
my $result = $session->export_data(
    type   => "dns_conf",
    path   => "/tmp/infoblox/dns_conf.tar.gz",
    member => "192.168.1.2" );

#Export DS records
my $result = $session->export_data(
    type   => "ds_records",
    zone   => "mydnsseczone.com",
    view   => "dnssec_view",
    path   => "/tmp/infoblox/ds_records.txt" );

#Export captive_portal_cert
my $result = $session->export_data(
    type => "captive_portal_cert",
    path => "/tmp/infoblox/captive_portal_cert.pem",
    dn   => 'CN="My CA Certificate",C="US"' );

#Export eap_ca_cert
my $result = $session->export_data(
    type => "eap_ca_cert",
    path => "/tmp/infoblox/eap_ca_cert.pem",
    dn   => 'CN="My CA Certificate",C="US"' );

#Generate and export a CSR For captive portal
my $result = $session->export_data(
    type     => "generate_captive_portal_csr",
    path     => "/tmp/infoblox/csr.pem",
```

```
    cn       => "My Signing Request",
    key_size =>  2048,
    org_unit => "My Dept.",
    org      => "My Company, Inc.",
    locality => "Santa Clara",
    state    => "California",
    member   => "192.168.1.2",
    country  => "US",
    email    => "me\@mycompany.com",
    comment  => "This is my certificate." );

#Generate and export a CSR For IF-MAP
my $result = $session->export_data(
    type     => "generate_ifmap_csr",
    path     => "/tmp/infoblox/csr.pem",
    cn       => "My Signing Request",
    key_size =>  2048,
    org_unit => "My Dept.",
    org      => "My Company, Inc.",
    locality => "Santa Clara",
    state    => "California",
    member   => "192.168.1.2",
    country  => "US",
    email    => "me\@mycompany.com",
    comment  => "This is my certificate." );

#Generate and export a self-signed admin certificate
my $result = $session->export_data(
    type       => "generate_self_signed_admin_cert",
    path       => "/tmp/infoblox/selfsigned.pem",
    cn         => "Self-Signed Admin Cert",
    member     => "192.168.1.2",
    days_valid => 365,
    key_size   => 2048,
    org_unit   => "My Dept.",
    org        => "My Company, Inc.",
    locality   => "Santa Clara",
    state      => "California",
    country    => "US",
    email      => "me\@mycompany.com",
    comment    => "This certificate is self-signed." );

#Generate and export a self-signed captive portal certificate
my $result = $session->export_data(
    type       => "generate_self_signed_captive_portal_cert",
    path       => "/tmp/infoblox/selfsigned.pem",
    cn         => "Self-Signed Captive Portal Cert",
    member     => "192.168.1.2",
    days_valid => 365,
    key_size   => 2048,
    org_unit   => "My Dept.",
    org        => "My Company, Inc.",
    locality   => "Santa Clara",
    state      => "California",
    country    => "US",
    email      => "me\@mycompany.com",
    comment    => "This certificate is self-signed." );

#Generate and export a self-signed IF-MAP certificate
my $result = $session->export_data(
    type       => "generate_self_signed_ifmap_cert",
    path       => "/tmp/infoblox/selfsigned.pem",
    cn         => "Self-Signed IF-MAP Cert",
    member     => "192.168.1.2",
    days_valid => 365,
    key_size   => 2048,
    org_unit   => "My Dept.",
    org        => "My Company, Inc.",
    locality   => "Santa Clara",
    state      => "California",
    country    => "US",
    email      => "me\@mycompany.com",
    comment    => "This certificate is self-signed." );

#Generate and export a CSR
My $result = $session->export_data(
```

```
    type     => "generate_csr",
    path     => "/tmp/infoblox/csr.pem",
    cn       => "My Signing Request",
    key_size =>  2048,
    org_unit => "My Dept.",
    org      => "My Company, Inc.",
    locality => "Santa Clara",
    state    => "California",
    member   => "192.168.1.2",
    country  => "US",
    email    => "me\@mycompany.com",
    comment  => "This is my certificate." );

#Export an IF-MAP certificate
my $result = $session->export_data(
    type   => "ifmap_cert",
    member => "192.168.1.2",
    path   => "/tmp/infoblox/ifmap_cert.pem");

#Export IF-MAP CIT XSD schema
my $result = $session->export_data(
    type   => "ifmap_cit_schema",
    cit    => $my_cit,
    path   => "/tmp/infoblox/my_cit.xsd" );

#Export an IF-MAP Federation certificate
my $result = $session->export_data(
    type => "ifmap_federation_cert",
    path => "/tmp/infoblox/ifmap_federation_cert.pem");

#Generate and export a self-signed IF-MAP Federation certificate
my $result = $session->export_data(
    type       => "generate_self_signed_ifmap_federation_cert",
    path       => "/tmp/infoblox/selfsigned.pem",
    cn         => "Self-Signed IF-MAP Federation Cert",
    member     => "192.168.1.2",
    days_valid => 365,
    key_size   => 2048,
    org_unit   => "My Dept.",
    org        => "My Company, Inc.",
    locality   => "Santa Clara",
    state      => "California",
    country    => "US",
    email      => "me\@mycompany.com",
    comment    => "This certificate is self-signed." );

#Generate and export a CSR for the IF-MAP Federation service
my $result = $session->export_data(
    type     => "generate_ifmap_federation_csr",
    path     => "/tmp/infoblox/csr.pem",
    cn       => "My Signing Request",
    key_size =>  2048,
    org_unit => "My Dept.",
    org      => "My Company, Inc.",
    locality => "Santa Clara",
    state    => "California",
    member   => "192.168.1.2",
    country  => "US",
    email    => "me\@mycompany.com",
    comment  => "This is my certificate." );

#Export lease_history
$session->export_data(
    type   => "lease_history",
    path   => "/tmp/infoblox/lease_history.gz",
    start_time  => "2005/09/01 00:00:00",
    end_time    => "2005/09/01 11:59:59" );

#Export the audit log
$session->export_data(
    type   => "log_files",
    path   => "/tmp/infoblox/auditlog.tar.gz",
    member => "192.168.1.2",
    node_type => "active",
    log_type => "audit");

#Export snmp_mibs
```

```
$session->export_data(
    type   => "snmp_mibs",
    path   => "/tmp/infoblox/snmp_mibs.tar.gz" );

#Export support_bundle
my $result = $session->export_data(
    type   => "support_bundle",
    path   => "/tmp/infoblox/supportBundle.tar.gz",
    member => "192.168.1.2" );

#Export support_bundle but exclude core files
my $result = $session->export_data(
    type   => "support_bundle",
    path   => "/tmp/infoblox/supportBundle.tar.gz",
    core_files => 'false',
    member => "192.168.1.2" );

#Export support_bundle but exclude log files
my $result = $session->export_data(
    type   => "support_bundle",
    path   => "/tmp/infoblox/supportBundle.tar.gz",
    log_files => 'false',
    member => "192.168.1.2" );

#Export serial number of all nodes
my $result = $session->export_data(
    type   => "serial_numbers",
    path   => "/tmp/infoblox/serial_nos.txt" );

#Export trust anchors
my $result= $session->export_data(
    type   => "trust_anchors",
    zone   => "mysignedzone.com",
    view   => "my_dnssec_view",
    path   => "/tmp/infoblox/trusted_keys.txt" );
```

fill_partial_object( )

Takes an object returned by a search method call with return_methods and fills it.

Parameter

object - Required. An object or a reference to an array of objects returned by a search method call using return_methods call, or from an AllRecords 'record' member. It is legal, but meaningless, to call this method for complete objects.

Returns

The method will return a complete object corresponding to the 'partial' object passed as a parameter. The object returned may be the same object (modified to have all methods initialized) or a new object. In the case of an error, undef will be returned and the appropriate status code will be set.

Example

```
# $partial_bind_object is an object returned by a search using return_methods
my $full_bind_object = $session->fill_partial_object($partial_bind_object);

# @partial_objects is an array of objects returned by a search using return_methods
my @full_objects = $session->fill_partial_object(\@partial_objects);
```

gen_tsig_key( )

Generates a TSIG key.

Parameter

none

Returns

The method returns a randomly generated TSIG key.

Example

```
my $tsig = $session->gen_tsig_key( );
```

get( )

Use this method to retrieve all the matching objects from the Infoblox appliance.

Parameter

object - Required. The Infoblox object to retrieve.

key reference - Required. The attribute and value that refer to the desired `object(s)` to retrieve. See individual objects for available key references.

Returns

Return an array in list context containing all matching objects. For failures and objects not found, it returns an empty array in list context and undefined in scalar context. When an operation fails, the error code and error message are reported in the `status_code()` and `status_detail()` attributes.

Example

```
#Getting an A record from the Infoblox appliance.
my @retrieved_objs = $session->get(
    object => "Infoblox::DNS::Record::A",
    name   => "bind_a.domain.com"  );

unless (@retrieved_objs) {
    die("Get DNS record A failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
```

import_data( )

Use this method to upload data to the Infoblox appliance from the local system.

Parameter

path - Required. The complete path of the file to import.

type - Required. The data type to import. Following is a list of available data types for import:

```
 +-------------------------------------------------------------------------------------------------------------+
 | admin_cert       - Import a certificate signed by a CA for a certificate signing request previously given by a    |
 |                    member. The input file should be in "pem" format.                                            |
 | member           - Required. The member to which you want to upload the certificate.                            |
 +-------------------------------------------------------------------------------------------------------------+
 | backup         - Restore an existing database backup. Supported file type is "tar.gz". The following parameters   |
 |                  are also available:                                                                            |
 | force          - Optional. Force restore a database that was backed up from a different grid. Default is "false". |
 | keep_grid_ip   - Optional. Used with the force restore option to retain the current grid IP address and ignore    |
 |                  the IP address from the backup. Default is "true".                                             |
 +-------------------------------------------------------------------------------------------------------------+
 | captive_portal_cert   - Import a server certificate for the captive portal.                                    |
 |                         Supported file type is "pem".                                                           |
 | member                - Required. The captive portal to which you want to upload the certificate.               |
 +-------------------------------------------------------------------------------------------------------------+
 | captive_portal_file   - Import a file to be used by the captive portal.                                        |
 |                         Supported file types are .jpg, .gif and .png (when cp_type is one of 'logo', 'footer' or  |
 |                         'header') or .txt (when cp_type is set to 'aup')                                        |
 | member                - Required. The captive portal to which you want to upload the file.                      |
 | cp_type               - Required. Indicates how the file is used in the captive portal. Valid values are:       |
 |                                     'logo', 'footer', 'header', 'aup'.                                          |
 +-------------------------------------------------------------------------------------------------------------+
 | clone          - For cloned data from the backup of another system. Supported file type is "tar.gz".            |
 +-------------------------------------------------------------------------------------------------------------+
 | csv              - Import DNS and DHCP data via CSV files. See export_data/csv for the supported object types.   |
 |                    For more information about the CSV file format please refer to the Infoblox
```

```
CSV Import        |
 |                   Reference.
|
 |
|
 | operation         - Optional. Valid values are "insert" or "update". The default is "insert".
|
 | override          - Optional. Valid values are "override" or "merge". The default is "override".
|
 | separator         - Optional. CSV separator character. Valid values are 'comma', 'semicolon',
'space' or 'tab'.   |
 |                     The default is 'comma'.
|
 | continue_on_error - Optional. Continue import in case of errors. Valid values are "true" or
"false".             |
 |                     The default is "false". If the value is "true", the import continues
even if there  |
 |                     are errors. Note that file format errors are reported immediately.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | dhcp_expert_mode_config - Import a custom DHCP configuration file from the local system to the
Infoblox           |
 |                           appliance. Supported file type is "dhcpd.conf". The following parameter
is also        |
 |                           available:
|
 | member                  - Required. The member to which you want to upload the DHCP configuration
expert mode     |
 |                           file.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | ds_records     - Import DS records from a file. The file format is either a DSset (DS records) or
a keyset        |
 |                  (DNSKEY records).
|
 | zone           - Required.  The zone that will contain the records.
|
 |                  Specify either the FQDN of the zone ore the Infoblox::DNS::Zone object.
|
 | view           - Optional. The DNS view that contains the zone. Specify either the DNS view name
or the           |
 |                  Infoblox::DNS::View object. The default is the "default" view.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | eap_ca_cert    - Import an Extensible Authentication Protocol Certificate Authority (EAP CA)
certificate.        |
 |                  Supported file type is "pem".
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | ifmap_cert       - Import a certificate signed by a CA for a certificate signing request
previously given by a     |
 |                    member for the IF-MAP service. The input file must be in "pem" format.
|
 | member           - Required. The member to which you want to upload the certificate.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | ifmap_cit_schema - Import a custom identifier type. The input file must be an XSD schema for the
|
 |                    custom identifier.
|
 | display_name     - Required. The name that is displayed for this CIT.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | ifmap_federation_cert - Import the CA-signed certificate for the IF-MAP Federation service. The
certificate       |
 |                    corresponds to the CSR previously generated by a member. The input file must be
in "pem" format|
 | member           - Required. The member to which you want to upload the certificate.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | keytab         - Import a keytab file that contains a host/key pair for the canonical name of the
host for        |
 |                  GSS-TSIG authentication. Supported file type is keytab file.
|
 | service        - "dns" or "dhcp". Default is "dns".
|
 | virtual_ip     - Either the virtual IP address or hostname of the node to which the file is
imported.  This is    |
 |                  a required parameter if service is 'dns'.
|
 +-------------------------------------------------------------------------------------------
---------------------------+
 | leases           - Import DHCP leases from the local system to the Infoblox appliance. Supported
file type is     |
 |                    "dhcpd.leases". The following parameters are also available:
|
 | format           - Optional. The format of the data file. Only "ISC" is supported. Default is
"ISC".             |
 | lease_precedence - Indicates whether existing leases are overwritten. Valid values are "keep-
previous",           |
```

```
 |                    "replace-previous", and "keep-newest". Default is "keep-newest".
|
 | network_view     - Optional. The name of the network view from which the lease data is imported.
|
 |                    The default value is the system-defined default network view.
|
 +-------------------------------------------------------------------------------------------
--------------------------+
 | leases6          - Import IPv6 DHCP leases from the local system to the Infoblox appliance.
Supported file type is|
 |                    "dhcpd.leases". The following parameters are also available:
|
 | format           - Optional. The format of the data file. Only "ISC" is supported. Default is
"ISC".               |
 | lease_precedence - Indicates whether existing leases are overwritten. Valid values are "keep-
previous",            |
 |                    "replace-previous", and "keep-newest". Default is "keep-newest".
|
 | network_view     - Optional. The name of the network view from which the lease data is imported.
|
 |                    The default value is the system-defined default network view.
|
 +-------------------------------------------------------------------------------------------
--------------------------+
 | lease_history  - Import the DHCP lease history. Supported file type is "gz".
|
 +-------------------------------------------------------------------------------------------
--------------------------+
 | tftp_file      - Import file from the local system to a TFTP directory on the Infoblox appliance.
All file types  |
 |                  are supported. The following parameter is also available:
|
 | directory      - Required. The TFTP directory in which the file is stored.
|
 +-------------------------------------------------------------------------------------------
--------------------------+
 | discovery_csv  - Import the file from the local system to the Infoblox appliance. Supported file
type is a CSV    |
 |                  file using comma as field sepearator. The following parameter is also available:
|
 | network_view   - Optional. The name of the network view from which the CSV data is imported.
|
 |                  The default value is the system-defined default network view.
|
 | merge_data     - The flag indicates whether the GM sorter replaces or merges the discovered data
with existing    |
 |                  data. The valid values are "true" and "false". The default value is "true".
|
 +-------------------------------------------------------------------------------------------
--------------------------+
```

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
#import backup
$session->import_data(
    type  => "backup",
    force => "false",
    path  => "/tmp/infoblox/database.tar.gz" );

#csv import
$session->import_data(
    type  => "csv",
    continue_on_error => 'true',
    operation => 'update',
    override => 'override',
    path  => "/tmp/infoblox/import_data.csv" );

#import clone
$session->import_data(
    type => "clone",
    path => "/tmp/infoblox/clone_database.tar.gz" );

#import dhcp_expert_mode_config
$session->import_data(
    type   => "dhcp_expert_mode_config",
    member => "192.168.1.2",
    path   => "/tmp/infoblox/dhcpd.conf" );

#import captive_portal_cert
$session->import_data(
    type => "captive_portal_cert",
    path => "/tmp/infoblox/eap_ca_cert.pem",
    member => "192.168.1.2" );
```

```
#import captive_portal_file
$session->import_data(
    type => "captive_portal_file",
    path => "/tmp/infoblox/logo_file.jpg",
    member => "192.168.1.2",
    cp_type => "logo" );

#import eap_ca_cert
$session->import_data(
    type => "eap_ca_cert",
    path => "/tmp/infoblox/eap_ca_cert.pem" );

#import the  IF-MAP certificate
$session->import_data(
    type => "ifmap_cert",
    path => "/tmp/infoblox/ifmap_cert.pem" );

#Import IF-MAP CIT
$session->import_data(
    type => "ifmap_cit_schema",
    display_name => "Location",
    path => "/tmp/infoblox/location_cit.xsd" );

#Import the IF-MAP Federation certificate
$session->import_data(
    type => "ifmap_federation_cert",
    path => "/tmp/infoblox/ifmap_federation_cert.pem" );

#import keytab
$session->import_data(
    type => "keytab",
    path => "/tmp/infoblox/keytab.key",
    virtual_ip => '192.168.1.2' );

$session->import_data(
    type => "keytab",
    service => "dhcp", # default is "dns"
    path => "/tmp/infoblox/keytab.key",
    virtual_ip => '192.168.1.2' );

#import leases
$session->import_data(
    type             => "leases",
    format           => "ISC",
    path             => "/tmp/infoblox/dhcpd.leases",
    lease_precedence => "keep-newest" );

#import leases from a non-default network view
$session->import_data(
    type             => "leases",
    format           => "ISC",
    path             => "/tmp/infoblox/dhcpd.leases",
    lease_precedence => "keep-newest",
    network_view     => "netview" );

#import lease_history
 $session->import_data(
    type => "lease_history",
    path => "/tmp/infoblox/lease_history.gz" );

#import tftp_file
$session->import_data(
    type      => "tftp_file",
    directory => "/",
    path      => "/tmp/infoblox/voip_settings.txt" );

#import discovery CSV file in a given network view
$session->import_data(
    type             => "discovery_csv",
    network_view     => "netview" );

#import DS records
$session->import_data(
    type             => "ds_records",
    path             => "/tmp/infoblox/ds_records.txt",
    zone             => "myzone.test.com",
    view             => "dns_view", );
```

join_grid( )

Join an Infoblox appliance to an existing grid.

Parameter

grid_name - Required. The name of the grid.

master - Required. The virtual IP address of the grid master.

shared_secret - Required. The shared secret string of the grid.

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
my $result = $session->join_grid(
    grid_name     => "infoblox",
    master        => "192.168.1.2",
    shared_secret => "test" );
```

modify( )

Use this method to modify an object in the Infoblox appliance.

To modify a specific object, use `get()` or `search()` to retrieve the specific object, make the desired modifications to this object, and then submit it for modification.

Parameter

The object to be modified in the Infoblox appliance (e.g.,Infoblox::DNS::Record::A). See individual objects for the necessary parameters.

Parameter

scheduled_at - Optional. The scheduled date and time in ISO 8601 extended format (e.g. 2008-10-21T10:56:00Z or 2008-10-21T10:56:00.001Z). See details.

Returns

The method returns true when the modification succeeds, and returns false when the operation fails. When an operation fails, the error code and error message are reported in the `status_code()` and `status_detail()` attributes.

Example

```
# Use get() to retrieve the object.
my @retrieved_obj = $session->get(
    object => "Infoblox::DNS::Record::A",
    name   => "bind_a.domain.com" );
# select the specific object.
my $bind_a = $retrieved_obj[0];
# Use object method to modify the comment.
$bind_a->comment("this is a modified comment");
# Submit modification
my $response = $session->modify( $bind_a );
```

new_cursor( )

Use this method to return a cursor object. See Infoblox::Cursor for more information.

network_discovery_control( )

Use this method to control the network discovery task

Parameter

action - Required. Valid values are "START", "PAUSE", "RESUME", and "END".

scheduled_at - Optional. Scheduled time in ISO 8601 extended format for date and time (e.g. 2008-10-21T10:56:00Z or 2008-10-21T10:56:00.001Z). Note that this can only be used with the "START" action.

Returns

The method returns true when the operation succeeds, and returns false when the operation fails. When an operation fails, the error code and error message are reported in the `status_code()` and `status_detail()` attributes.

Example

```
# start network discovery
my $response = $session->network_discovery_control(
      action => 'START');

unless($response) {
   die("Start network discovery failed");
}
```

remove( )

Use this method to remove an object from the Infoblox appliance.

To remove a specific object, first use `get()` or `search()` to retrieve the specific object, and then submit this object for removal.

Parameter

The object to be removed from the Infoblox appliance (e.g.,Infoblox::DNS::Record::A). See individual objects for the necessary parameters.

Parameter

scheduled_at - Optional. Scheduled time in ISO 8601 extended format for date and time (e.g. 2008-10-21T10:56:00Z or 2008-10-21T10:56:00.001Z). See details.

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
# Use get() to retrieve the object.
my @retrieved_obj = $session->get(
    object => "Infoblox::DNS::Record::A",
    name   => "bind_a.domain.com" );
# Select the specific object.
my $bind_a = $retrieved_obj[0];
# Submit for removal
my $response = $session->remove( $bind_a );
```

remove_data( )

Use this method to remove data from the Infoblox appliance or purge DHCP IF-MAP data.

Parameter

type - Required. The type of data to remove. Following is a list of data types that can be removed:

```
 ---------------------------------------------------------------------------------------------------
 | keytab                    - Remove a keytab file that contains a host/key pair for the canonical
name of the host      |
 |                             for GSS-TSIG authentication. The following parameters are also
available:                   |
 | service                   - Optional. Either "dns" or "dhcp". Default is "dns".
|
 | virtual_ip                - Either the virtual IP address or hostname of the appliance to which the
file is imported.  |
 |                             This is a required parameter if the service is 'dns'.
|
 ---------------------------------------------------------------------------------------------------
 | nac_auth_cache            - Purge the NAC authentication cache.
|
 | member                    - Required. The IPv4 address of the member on which the NAC
authentication cache is cleared. |
 | mac_addr                  - Optional. The MAC address for which the authentication result in the
NAC authentication    |
 |                             cache is to be cleared. if this field is not present, the entire cache
will be flushed.    |
 ---------------------------------------------------------------------------------------------------
 | ifmap_dhcp_data           - Sends a purge request that removes any IF-MAP data published by the
DHCP IF-MAP client     |
 |                             running on this member. The following parameters are also available:
|
 |
|
 | member                    - Required. The IP address of the grid member.
|
 ---------------------------------------------------------------------------------------------------
```

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
# purge DHCP IF-MAP data published by member 1.2.3.4
$session->remove_data(
    "type"      => "ifmap_dhcp_data",
    "member" => "1.2.3.4");

# delete imported DNS GSS-TSIG keytab data
$session->remove_data(
     "type"        => "keytab",
     "service"   => "dns",
     "virtual_ip"  => "192.168.1.2");

# purge the entire NAC authentication cache for member 1.2.3.4
$session->remove_data(
    "type"      => "nac_auth_cache",
    "member" => "1.2.3.4");

# Clear authentication data for MAC address a1:a1:a1:a1:a1:a1 from the NAC authentication cache for
member 1.2.3.4
$session->remove_data(
    "type"          => "nac_auth_cache",
    "member"    => "1.2.3.4",
    "mac_addr" => "a1:a1:a1:a1:a1:a1");
```

remove_eap_ca_cert( )

Remove an Extensible Authentication Protocol Certificate Authority (EAP CA) certificate.

Parameter

dn - Required. Distinguished Name (DN) of the Certificate Authority (CA) certificate you want to remove.

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
my $result = $session->remove_eap_ca_cert(
    dn => 'CN="My Server Certificate",C="US"' );
```

remove_tftp_file( )

Remove a file from a TFTP directory on the Infoblox appliance.

Parameter

directory - Required. The parent directory of the file in TFTP storage.

name - Required. The name of the file.

Returns

The method returns true when the modification succeeds, and returns false when the operation fails.

Example

```
my $result = $session->remove_tftp_file(
    directory => "/",
    name      => "voip_settings.txt" );
```

reset_discovery_timestamp( )

This method is deprecated and is provided only to maintain backward compatibility with previous releases.

Calling this method will perform the same action as calling the **reset_discovery_data()** method.

reset_discovery_data( )

Use this method to reset the discovery related attributes for an IP address.

The following attributes are reset:

```
os
netbios
first_discovered
last_discovered
network_component_type
network_component_name
network_component_description
network_component_ip
network_component_port_number
network_component_port_name
network_component_port_description
port_vlan_name
port_vlan_number
port_speed
port_duplex
port_status
port_link_status
discovered_name
discoverer
```

Parameter

type - Required. Valid values are "GLOBAL", "NETWORK", and "IP_ADDRESS".

network - Optional. Required when type is "NETWORK" or "IP_ADDRESS".

network_view - Optional. The network view to which the reset operation will be limited. The default value is the system-defined default network view.

ip_address - Optional. Required when type is "IP_ADDRESS".

Returns

The method returns true when the operation succeeds, and returns false when the operation fails. When an operation fails, the error code and error message are reported in the `status_code()` and `status_detail()` attributes.

Example

```
# type is GLOBAL
my $response = $session->reset_discovery_data(
       type => 'GLOBAL');

unless($response) {
   die("reset discovery datafailed");
}

# with a non-default network view
my $nview = Infoblox::DHCP::View->new(
       name => 'netview',
);
my $response = $session->reset_discovery_data(
       type => 'GLOBAL',
       network_view => $nview);

unless($response) {
   die("reset discovery data failed");
}

# type is NETWORK
$response = $session->reset_discovery_data(
       type => 'NETWORK',
       network => '10.0.0.0/24');

unless($response) {
   die("reset discovery data failed");
}

# type is IP_ADDRESS
$response = $session->reset_discovery_data(
       type => 'IP_ADDRESS',
       network => '10.0.0.0/24',
       ip_address => '10.0.0.10');

unless($response) {
   die("reset discovery data failed");
}
```

restart( )

Restart a service on the Infoblox appliance.

Parameter

member - Optional. The address of the member that restarts service. Default is all members.

force_restart - Optional. Valid values are "true" or "false".

delay_between_members - Optional. Valid only when a member is not specified. Valid value is the number of seconds. Default value is 0 seconds.

scheduled_at - Optional. Scheduled time in ISO 8601 extended format for date and time (e.g. 2008-10-21T10:56:00Z or 2008-10-21T10:56:00.001Z).

Returns

The method returns true when the modification succeeds, and returns false when the operation fails. If no attribute is specified, the method returns the pending schedule.

Example

```
my $result = $session->restart(
    delay_between_members => 4,
    member                => "192.168.1.2");
```

restart_status( )

Use this method to retrieve all the services that need to restart.

Parameter

None

Returns

The method returns an array of strings for all the services that need to restart ('dhcp', 'dns').

Example

```
my @restart_list = $session->restart_status();
```

search( )

Use this method to search and retrieve all the matching objects from the Infoblox appliance.

Parameter

object - Required. The Infoblox object to retrieve.

return_methods - Optional. The member fields to return.

key reference - Required. The attribute and value that refer to the desired `object(s)` to retrieve. Many key references are given as regular expressions. See individual objects for available key references.

Returns

Return an array in list context containing all matching objects. For failures or objects not found, it returns an empty array in list context and undefined in scalar context. When an operation fails, the error code and error message are reported in the `status_code()` and `status_detail()` attributes.

If the return_methods parameters are passed, only the specified members will be set in the returned objects. Note that these objects cannot be used directly for modify( ) calls; fill_partial_object needs to be called for them first.

Member methods that represent actions/operations rather than values (e.g., Infoblox::DHCP::NetworkContainer->resize()) cannot be specified as return_methods.

Example

```
#Searching for all A records under "domain.com" from the Infoblox appliance.
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::A",
    name   => '.*\.domain\.com' );

unless (@retrieved_objs) {
    die("Search DNS record A failed: ",
       $session->status_code() . ":" . $session->status_detail());
}

#Searching for all A records under "domain.com" from the Infoblox appliance and returning only their
addresses and TTL's.
```

```
my @retrieved_objs = $session->search(
    object => "Infoblox::DNS::Record::A",
    name   => '.*\.domain\.com',
    return_methods => ['ttl','ipv4addr']
);

unless (@retrieved_objs) {
    die("Search DNS record A failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
```

server_version( )

Check the current server version.

Parameter

none

Returns

The method returns the current Infoblox appliance version.

Example

```
my $version = $session->server_version( );
```

set_dhcp_failover_partner_down( )

Set a DHCP failover association peer to the Partner Down state.

Parameter

dhcp_failover - Required. The name of the DHCP failover association with the peer that you are setting to the Partner Down state.

Parameter

peer_type - Required. The peer that you are setting to the Partner Down state. Valid values are 'primary' or 'secondary'.

Returns

The method returns true when the operation succeeds, and returns false when the operation fails.

Example

```
my $result = $session->set_dhcp_failover_partner_down(
   dhcp_failover => 'test_failover',
   peer_type     => 'primary'
);
```

set_dhcp_failover_secondary_recover( )

Set the secondary peer to the Recover state.

Parameter

dhcp_failover - Required. The name of the DHCP failover association with the secondary peer that you are setting to the Recover state.

Returns

The method returns true when the operation succeeds, and returns false when the operation fails.

Example

```
my $result = $session->set_dhcp_failover_secondary_recover(
   dhcp_failover => 'test_failover'
);
```

status_code( )

Get the status code of the last function called.

Parameter

none

Returns

The method returns the status code of the last function/operation called.

Example

```
my $status_code = $session->status_code( );
```

status_detail( )

Get the status detail of the last function called.

Parameter

none

Returns

The method returns the status detail of the last function/operation called.

Example

```
my $status_detail = $session->status_detail( );
```

SAMPLE CODE

#Create an Infoblox::Session object

```
#PROGRAM STARTS: Include all the modules that will be used
use strict;
use Infoblox;

#Create a session to the Infoblox appliance
my $session = Infoblox::Session->new(
    master   => "192.168.1.2",
    username => "admin",
    password => "infoblox"
);
if ($session->status_code()) {
   die("Construct session failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Session created successfully\n";
```

#Add a DNS zone object

```
#Construct the zone
my $zone = Infoblox::DNS::Zone->new(name => "domain.com");
unless ($zone) {
   die("Construct zone failed: ",
       $session->status_code() . ":" . $session->status_detail());
}
print "Zone object created successfully\n";

#Add the zone to the Infoblox appliance
$session->add($zone)
    or die("Add zone failed: ",
        $session->status_code() . ":" . $session->status_detail());
print "Zone added successfully\n";
```

#Restart service

```
$session->restart(
     "member" => "192.168.1.2",
     "when"   => "03/20/2007 01:00:00",
     "time_zone" => "(UMT - 6:00) Central Time (US and Canada)"
);

#To cancel the previous restart schedule.

 $session->restart("cancel" => "true");
```

#Get restart services

```
my @services   $session   restart status
```

```
              =          ->              ();

#Get zone object

 my @retrieved_objs = $session->get(
     object    => "Infoblox::DNS::Zone",
     name      => "domain.com" );
 my $object = $retrieved_objs[0];
 unless ($object) {
     die("Get Zone object failed: ",
         $session->status_code() . ":" . $session->status_detail());
 }
 print "Get DNS zone object found at least 1 matching entry\n";

#Remove zone object

 $session->remove( $object )
   or die("Remove DNS zone failed: ",
         $session->status_code() . ":" . $session->status_detail());
 print "DNS Zone object removed successfully \n";

#Show the current version

 my $version = $session->server_version();
 print "Current server version is $version\n";

#Generate a TSIG key

 my $tsig = $session->gen_tsig_key();
 print "Generated a TSIG key: $tsig\n";

#Show status_code and status_detail

 my $status_code = $session->status_code();
 my $status_detail = $session->status_detail();
 print "Previous operation status_code is: $status_code\n";
 print "Previous operation status_detail is: $status_detail\n";

#clear unmanaged address

 # Creating new network and checking
 my $network = Infoblox::DHCP::Network->new(network => "10.0.0.0/24");
 unless($network) {
    die("Create network object failed.");
 }

 # Adding into Session and testing
 $session->add($network)
    or die("Add network failed.");

 # Creating a new network view
 my $network_view = Infoblox::DHCP::View->new(name => "netview");
 unless($network_view) {
    die("Create network view object failed.");
 }
 $session->add($network_view)
    or die("Add network view failed.");

 # Creating a second network in the network view
 my $network2 = Infoblox::DHCP::Network->new(
                 network => "10.0.0.0/24",
                 network_view => $network_view);
 unless($network2) {
    die("Create second network object failed.");
 }
 $session->add($network2)
    or die("Add second network object failed.");

 # type is GLOBAL
 my $response = $session->clear_unmanaged(
    type => 'GLOBAL');

 unless($response) {
    die("Clear unmanaged failed");
 }

 my $response = $session->clear_unmanaged(
```

```
   type => 'GLOBAL',
   network_view => $network_view);

unless($response) {
   die("Clear unmanaged failed");
}
# type is NETWORK
$response = $session->clear_unmanaged(
   type => 'NETWORK',
   network => '10.0.0.0/24');

unless($response) {
   die("Clear unmanaged failed");
}

$response = $session->clear_unmanaged(
   type => 'NETWORK',
   network => '10.0.0.0/24',
   network_view => $network_view);

unless($response) {
   die("Clear unmanaged failed");
}

# type is IP_ADDRESS
$response = $session->clear_unmanaged(
   type => 'IP_ADDRESS',
   network => '10.0.0.0/24',
   ip_address => '10.0.0.10');

unless($response) {
   die("Clear unmanaged failed");
}

$response = $session->clear_unmanaged(
   type => 'IP_ADDRESS',
   network => '10.0.0.0/24',
   network_view => $network_view,
   ip_address => '10.0.0.10');

unless($response) {
   die("Clear unmanaged failed");
}
```

#reset discovery data

```
# type is GLOBAL
my $response = $session->reset_discovery_data(
   type => 'GLOBAL');

unless($response) {
   die("reset discovery data failed");
}

my $response = $session->reset_discovery_data(
   type => 'GLOBAL',
   network_view => $network_view);

unless($response) {
   die("reset discovery data failed");
}

# type is NETWORK
$response = $session->reset_discovery_data(
   type => 'NETWORK',
   network => '10.0.0.0/24');

unless($response) {
   die("reset discovery data failed");
}

$response = $session->reset_discovery_data(
   type => 'NETWORK',
   network => '10.0.0.0/24',
   network_view => $network_view);

unless($response) {
```

```
   die("reset discovery data failed");
}

# type is IP_ADDRESS
$response = $session->reset_discovery_data(
   type => 'IP_ADDRESS',
   network => '10.0.0.0/24',
   ip_address => '10.0.0.10');

unless($response) {
   die("reset discovery data failed");
}

$response = $session->reset_discovery_data(
   type => 'IP_ADDRESS',
   network => '10.0.0.0/24',
   network_view => $network_view,
   ip_address => '10.0.0.10');

unless($response) {
   die("reset discovery data failed");
}
```

#Control the network discovery task

```
# stop network discovery
my $response = $session->network_discovery_control(
       action => 'END',
       network => '10.0.0.0/24');

unless($response) {
   die("Stop network discovery failed");
}
```

#Clean up

```
#Remove the network
$network = $session->get(
             object => 'Infoblox::DHCP::Network',
             network => '10.0.0.0/24',
             network_view => 'default');
unless($network) {
   die("network '10.0.0.0/24' couldn't be retrieved");
}
$session->remove($network)
   or die("Remove DHCP network failed: ",
              $session->status_code() . ":" . $session->status_detail());
print "DHCP network object removed successfully \n";

#Remove the network view
# (network2 will be removed with the view)
$network_view = $session->get(
             object => 'Infoblox::DHCP::View',
             name => 'netview');
unless($network_view) {
   die("network view 'netview' couldn't be retrived");
}
$session->remove($network_view)
   or die("Remove DHCP network view failed: ",
              $session->status_code() . ":" . $session->status_detail());
print "DHCP network view object removed successfully \n";
```

SCHEDULED OBJECT USAGE DETAILS

The following are the object types for which you can schedule tasks:

```
Infoblox::DHCP::FixedAddr
Infoblox::DHCP::NetworkContainer
Infoblox::DHCP::IPv6NetworkContainer
Infoblox::DHCP::Network
Infoblox::DHCP::IPv6Network
Infoblox::DHCP::SharedNetwork
Infoblox::DHCP::Range
Infoblox::DHCP::RoamingHost
Infoblox::DNS::BulkHost
Infoblox::DNS::Host
Infoblox::DNS::Record::A
Infoblox::DNS::Record::AAAA
```

```
Infoblox::DNS::Record::CNAME
Infoblox::DNS::Record::DNAME
Infoblox::DNS::Record::MX
Infoblox::DNS::Record::PTR
Infoblox::DNS::Record::SRV
Infoblox::DNS::Record::TXT
Infoblox::DNS::SharedRecord::A
Infoblox::DNS::SharedRecord::AAAA
Infoblox::DNS::SharedRecord::MX
Infoblox::DNS::SharedRecord::SRV
Infoblox::DNS::SharedRecord::TXT
Infoblox::DNS::Zone
Infoblox::DNS::SRG
Infoblox::DNS::View
```

The following are the functions for which you can schedule tasks:

```
restart
network_discovery_control
```

To allow limited-access admins to use the scheudling feature, add the permission with resource_type = "Scheduling" to the appropriate admin group (see Infoblox::Grid::Admin::Permission).

AUTHOR

Infoblox Inc. http://www.infoblox.com/

SEE ALSO

Infoblox::DNS::Record::A, Infoblox::DNS::Zone, Infoblox::DNS::View, Infoblox::DHCP::FailOver, Infoblox::DHCP::Network, Infoblox::DHCP::NetworkContainer ,Infoblox::DHCP::View, Infoblox::Grid::GlobalSmartFolder, Infoblox::Grid::PersonalSmartFolder, Infoblox::Grid::ScheduledTask

---

COPYRIGHT

Copyright (c) 1999-2011 Infoblox Inc.

Appendix A CSV File Format

You can use the *import_data* method in the Infoblox::Session class of the Infoblox API to upload discovery files from PortIQ appliances. The files must be in CSV (comma separated values) format as follows:

- All timestamps are in UTC (Coordinated Universal Time). The format is: *YYYY-MM-DD HH:MM:SS*.
- The header line of the CSV file accepts the following columns:
  - ip_address — The discovered IPv4 address. Required.
  - mac_address — A valid MAC address. Optional.
  - last_discovered_timestamp — UTC timestamp of the last time the discoverer saw the IP address. Required.
  - first_discovered_timestamp — UTC timestamp of the first time the discoverer saw the IP address. Optional.
  - netbios_name — The NETBIOS name of the device of the discovered IP address. String type. Maximum size is 15 characters. Optional.
  - os — The OS of the device of the discovered IP address. String type. Maximum size is 256 characters. Optional.
  - network_component_type — The type of network component connected to the device of the discovered IP address; for example, switch, router, or other. Optional. String type. Maximum size is 32 characters.
  - network_component_name — Name of the network component connected to the discovered IP address device. Optional. String type. Maximum size is 64 characters.
  - network_component_description — Description of the network component connected to the discovered IP address device. Optional. String type. Maximum size 256 characters.
  - network_component_ip — IP address of the network component. Optional. String type. IPv4 address format.
  - network_component_port_number — Number of the network component port connected to the end device. Optional. Unsigned integer type. Range is 0 through 9999.
  - network_component_port_name — Name of the network component port connected to the end device. Optional. String type. Maximum size is 64 characters.
  - network_component_port_description — Description of the network component port connected to the end device. Optional. String type. Maximum size is 256 characters.
  - port_vlan_name — Name of the VLAN of the network component port. Optional. String type. Maximum size is 64 characters.
  - port_vlan_description — Description of the VLAN of the network component port. Optional. String type. Maximum size is 256 characters.
  - port_vlan_number — The number of the VLAN of the network component port. Optional. Unsigned integer type. Range is 0 through 9999.
  - port_speed — Speed setting on the network component port. Optional. String type. Valid values are: 10M, 100M, 1G, 10G, 100G, Unknown.

- port_duplex — Duplex setting on the network component port. Optional. String type. Valid values are `Full` and `Half`.
- port_status — The operational status of the network component port. Optional. String type. Valid values are `Up`, `Down`, and `Unknown`.
- port_link_status — The link status of the network component port. Indicates whether it is connected. Optional. String type. Valid values are: `Connected`, `Not Connected`, and `Unknown`.
- discovered_name — DNS name of the device associated with the discovered IP address. Optional. String type. Maximum size is 256 characters.
- discoverer — Specifies whether the IP address was discovered by a PortIQ appliance or NIOS discovery process. Optional. String type. Maximum size is 64 characters.

The following conventions apply to fields (header and data lines):

- Fields are separated by the field separator comma (,). Example: `field1,field2,field3`
- Fields can be quoted using double quotes. The actual data of a quoted field is all characters of the field minus the start and ending quotes. The following example is equivalent to the preceding example: `field1,"field2",field3`.
- Fields using the field separator as part of the data must be quoted. Any field separator within the double quotes of a field are considered part of the data. Example: `field1,"comma,field",field3`
- Fields using a double quote as part of the data must be double quoted and must double the data double quote. Following are examples:

  ```
  field1, "double""quote", field3
  field1, "ending doublequote""", field3
  field1, "double double quote""""", field3
  field1, "double quote"", coma in same field", field3
  ```
- The header line defines the order of columns and their respective expected content.
- The header line must contain at least the mandatory fields to be accepted. Columns containing optional fields can be omitted
- Field names in the header line are not case-sensitive.