Lucent Technologies
Bell Labs Innovations

VitalQIP®
Lucent DHCP Rules Manager (LDRM)

Version 2.0

Administrator Guide

190-409-066R6.2
Issue 1
August, 2005

Trademarks

Mandatory customer information

This information product does not contain any mandatory customer information.

Licenses

Java,

Jython,

Contents

About this document

Purpose

(LDRM) enables VitalQIP users to author custom rules for modifying fields and options of both incoming and outgoing DHCP packets.

How to use this information product

This manual is organized as follows:

Chapter 1: LDRM Overview	This chapter provides background about the prerequisites, architecture and interaction between LDRM and VitalQIP
Chapter 2: Installation	This chapter provides information on installing LDRM, including hardware and software prerequisites and installation instructions.
Chapter 3: Configuration	This chapter describes the settings you need to check and modify so LDRM works with VitalQIP.
Chapter 4: Rules	This chapter describes how to create and test LDRM rules. It contains variable documentation and examples of rules and test input packet files.
Chapter 5: Troubleshooting	This chapter describes some common problems you can encounter when creating rules, and provides suggested resolutions.

Conventions used

Table 1 lists the typographical conventions used throughout this manual.

Table 1 Typographical conventions

Convention	Meaning	Example
boldface	Names of items on screens. Names of commands and routines. Names of buttons you should click.	Select the **Client** check box. The **qip_getappllst** routine returns the entire list of existing applications. Click **OK**.
Arial boldface	Names of keys on the keyboard to be pressed.	Press **Enter** to continue.
courier font	Input that you should enter from your keyboard.	Run the following command: `c:\setup.exe`
<angle brackets>	Variables that you must substitute another value for.	***http://<VitalQIP_server_IP_address_or_name>***
italics	Manual names.	See the *VitalQIP Administrator Reference Manual* for more information.
Arial italic	Pathnames, filenames and email addresses	*c:\ldrm\conf\rules.jar*
Bold italic	Uniform Resource Locators (URLs) and emphasis	The VitalQIP Web site is ***http://qip.lucent.com.***
click	Click the left button on your mouse once.	To delete the object, click **Delete**.

Related information

The following documents are referenced in this manual:

- *VitalQIP Administrator Reference Manual* (part number: 190-409-042R6.2)

  This guide describes planning and configuring your network, information about the VitalQIP interface, advanced DNS and DHCP configurations, and troubleshooting.
- *VitalQIP Installation Guide* (part number: 190-409-043R6.2)

  This guide describes how to install the VitalQIP product.
- *VitalQIP User's Guide* (part number: 190-409-068R6.2)

  This guide describes how to set up and use the VitalQIP user interface on Windows and UNIX platforms.

Information product support

The Information Products & Training Group within Lucent offers cost-effective educational programs that cover the VitalQIP products and add-ons. Our offerings also include courses on the underlying technology for the VitalQIP products (for example, DNS and DHCP). Our classes blend presentation, discussion, and hands-on exercises to reinforce learning. Students acquire in-depth knowledge and gain expertise by practicing with our products in a controlled, instructor-facilitated setting. If you have any questions, please email us at *vitaleducation@lucent.com*.

Technical support

If you need assistance with VitalQIP, you can contact the Technical Assistance Center for your region. Contact information is provided in Table 2.

Table 2 Technical support information

Region	Address	Contact information
North, Central, and South America	Lucent Technologies 400 Lapp Road Malvern, PA 19355 USA	Phone: 1-866-LUCENT8 (582-3688) Option 5 Web: ***http://www.lucent.com/support***
Europe, Middle East, Africa, and China	Lucent Technologies Chiltern House Sterling Court Broad Lane Bracknell, RG12 9GU UK	Phone: 00 800 00 LUCENT or +353 1 692 4579 E-mail: *emeacallcenter@lucent.com* Web: ***http://www.lucent.com/suppor***t
Asia Pacific	Lucent Technologies Australia 68 Waterloo Rd North Ryde NSW 2113 Australia	Phone: 1800-458-236 (toll free from within Australia) (IDD) 800-5823-6888 (toll free from Asia Pacific - Hong Kong, Indonesia, South Korea, Malaysia, New Zealand, Philippines, Singapore, Taiwan, and Thailand) (613) 9614-8530 (toll call from any country) E-mail: *apactss@lucent.com*

How to order

Customers can order additional VitalQIP manuals online at ***http://www.cic.lucent.com/documents.html.***

How to comment

To comment on this information product, go to the Online Comment Form or email your comments to the Comments Hotline (*comments@lucent.com*).

□

1 LDRM Overview

Overview

Purpose

This section describes Lucent DHCP Rules Manager (LDRM).

Contents

This information presents the following topics.

☐

Introduction

LDRM enables VitalQIP users to author custom rules for modifying fields and options of both incoming and outgoing DHCP packets.

LDRM is an add-on product to Lucent’s VitalQIP product. VitalQIP without LDRM allows you to specify DHCP options by address range, userclass, and vendor class only. The DHCP server is configured through option templates associated with address scopes and client classes to assign option values.

With LDRM, the DHCP server can still be configured through option templates associated with address scopes and client classes to assign option values. If LDRM assigns values for the same options as are defined by the address scope or client classes, the LDRM assigned option values take precedence.

LDRM allows you to override most DHCP packet values, and to do so conditionally, that is, based on other DHCP packet values. For example, this enables you to set lease time on a client-provided hostname, or assign a subnet based on a relay agent option (option 82) circuit ID. You can require multiple conditions be true to override a packet value, or override multiple packet values with a single condition.

□

Prerequisites

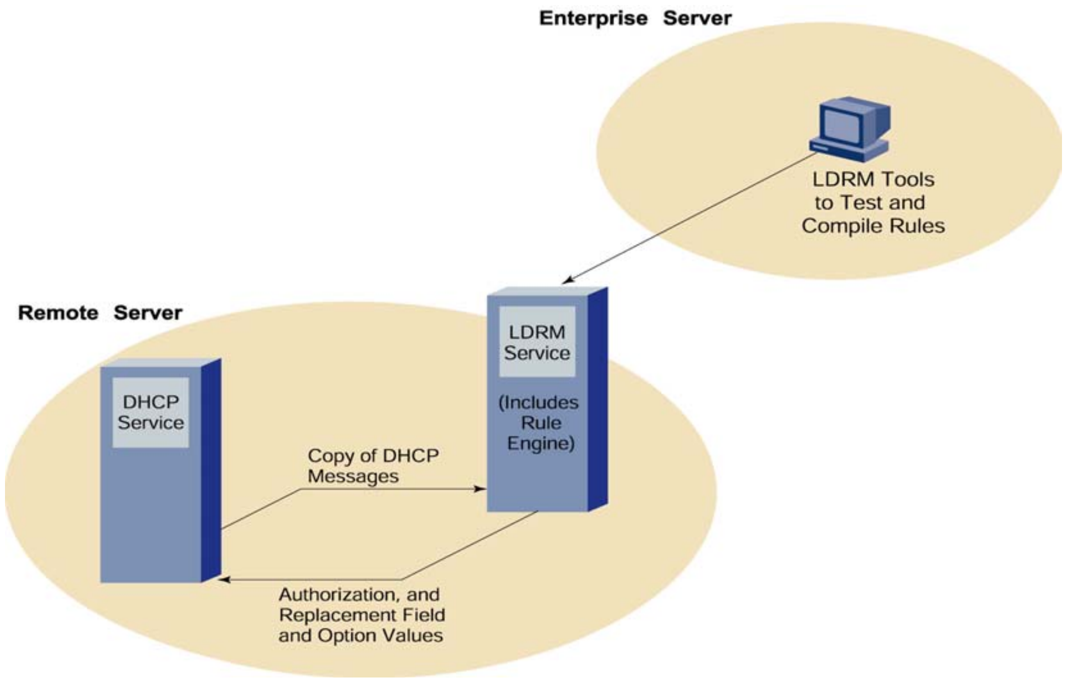
You must have a basic knowledge of Jython syntax before you begin using LDRM. See ***www.jython.org*** and select the documentation link for detailed Jython documentation.

☐

Architecture

LDRM is a VitalQIP service. It is based on Jython, an implementation of the high-level, dynamic, object-oriented language Python which is integrated with the Java platform. LDRM is installed on both enterprise servers and remote servers. The enterprise server is where you write and test rules, and compile the rules into the *rules.jar* file. After you create the *rules.jar* file, VitalQIP distributes it to the remote servers during DHCP File Generation, as shown in Figure 1:

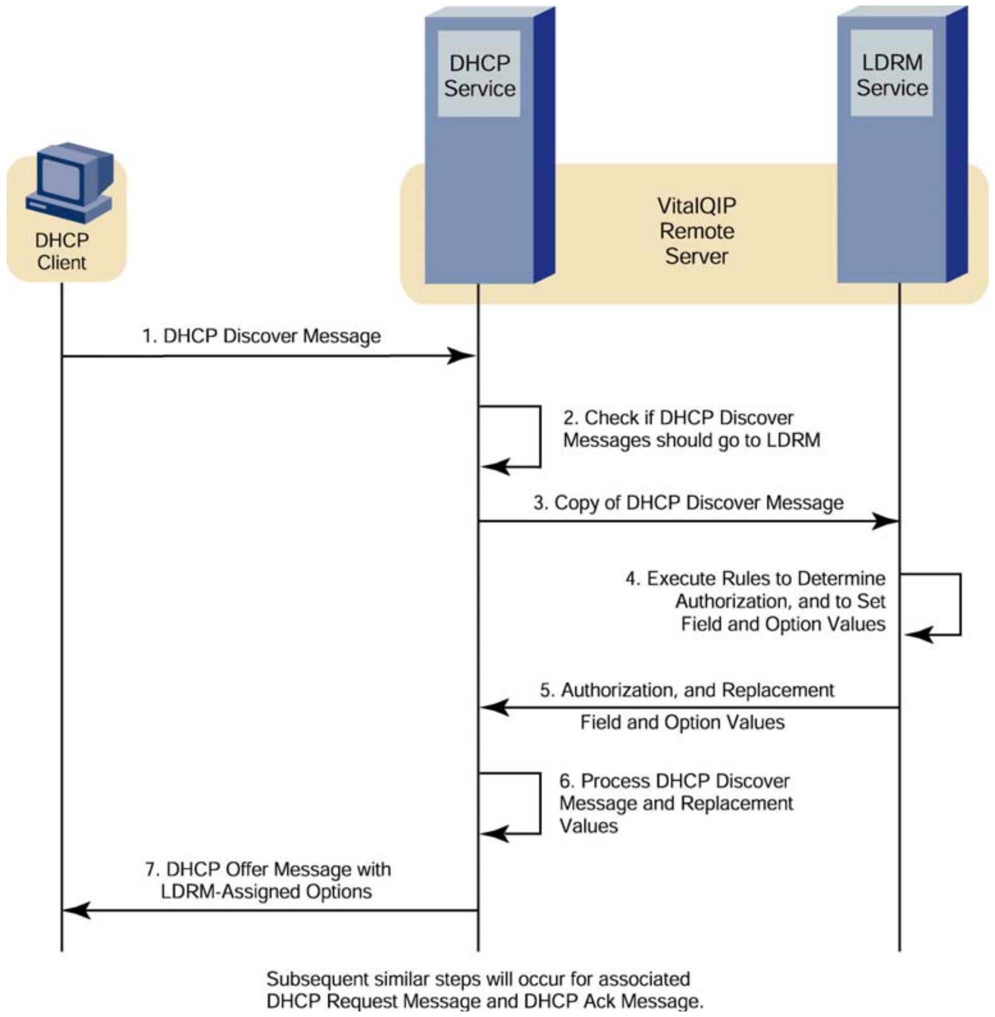
Figure 1 LDRM Architecture Overview



On the remote server, DHCP sends a copy of the packet to LDRM. LDRM checks the packet against the rules you have in place and sends the overridden values, if any, back to DHCP for continued processing.

The interface with the Lucent DHCP Service is through a new API callout library. The library supports the passing of incoming DHCP packets to LDRM as well as the specification of option values for the outgoing DHCP packets sent to DHCP clients. Figure 2 shows an example of LDRM dataflow:

Figure 2 Example LDRM dataflow


DHCP Service
LDRM Service
DHCP Client
VitalQIP Remote Server
1. DHCP Discover Message
2. Check if DHCP Discover Messages should go to LDRM
3. Copy of DHCP Discover Message
4. Execute Rules to Determine Authorization, and to Set Field and Option Values
5. Authorization, and Replacement Field and Option Values
6. Process DHCP Discover Message and Replacement Values
7. DHCP Offer Message with LDRM-Assigned Options
Subsequent similar steps will occur for associated DHCP Request Message and DHCP Ack Message.

When the DHCP client sends a Discover message to the DHCP Service, located on the VitalQIP Remote Server, the DHCP Service checks the message to see if it should be processed by LDRM. If LDRM should process it, the DHCP Service sends a copy of the DHCP Discover message to LDRM.

LDRM processes the copied packet and sets field and option values based on the LDRM rules. The replacement values, if any, are then returned to the DHCP Service. The DHCP Service processes the DHCP Discover message based on the replacement values, and returns a DHCP Offer message with the LDRM-assigned option values to the DHCP client.

API callouts

The Lucent DHCP server uses a series of API callouts to interface with the LDRM service. DHCP server policies determine which API callouts are actually invoked during the processing of DHCP packets. These policies are configured in the Server Profile, using VitalQIP, and must be consistent with the rules that are being executed in the LDRM service.

Callouts used

The callouts that you configure the Lucent DHCP server to invoke depend upon what you are attempting to achieve with the LDRM rules. The packet receipt API callout is utilized only for overriding the client hardware address (chaddr or MAC) and/or the client hostname values. This is particularly useful in a scenario where a proxy device may be attempting to obtain multiple leases for devices that it manages (for example, a proxy cradle device acquiring DHCP leases for wireless handheld scanner devices). When this callout is enabled, the Lucent DHCP server invokes the callout immediately following the receipt and parsing of every DHCP packet, regardless of the DHCP message type. The callout sends a copy of the packet to the LDRM service using TCP, and waits for a response. Upon receipt of an “authorized” response from the LDRM service, the DHCP server continues to process the packet, using the override values. When the response does not indicate “authorized”, the packet is dropped.

When the intent of the rules is to affect the configuration options that are sent to DHCP clients, the discover API and request API are utilized. This capability is useful when the standard DHCP configuration methods supported by VitalQIP do not provide enough flexibility for setting option values (for example, lease time value determined by combination of vendor class, user class and relay agent information). When these callouts are enabled, the Lucent DHCP server invokes the callout during the processing of the DHCPDISCOVER and DHCPREQUEST messages, respectively. Upon receipt of an “authorized” response from the LDRM service, the DHCP server constructs the DHCPOFFER and DHCPACK messages, using the override values provided by the LDRM service.

Finally, the bootp request API can be utilized to affect the configuration options that are sent to bootp clients in the bootp reply message. Also note that the packet receipt API can be used in combination with any of the other callouts described above.

□

Product application

LDRM can accomplish multiple DHCP message processing related functions. For example, after receiving a Discover, Request, or Bootp Request message from the DHCP Service, LDRM can respond by sending an array of overridden option values to the DHCP Service that it then sends to the client in a DHCP Offer or Ack message.

After the Lucent DHCP Service receives a message from a client, if appropriate, it forwards the message to LDRM. LDRM is co-located with the DHCP Service on the Remote Server. Then user-defined configurable rules in LDRM are executed (LDRM rules are written in the Jython scripting language). For instance, after receiving a message, LDRM can read one or more of several fields and options in the message and perform the following functions:

- Execute desired logic to override one or more of several fields and options in the message
- Set any of dozens of options to be used in the associated outgoing message
- Send the overriden values and the options for the DHCP service to use

After receiving this information, the DHCP Service processes the message and replacement values for it, and, if appropriate, sends a response back to the DHCP client using the options provided by LDRM.

For example, you can write LDRM rules to set a specific value for User Class (DHCP Option 77) and for Requested Lease Time (DHCP Option 51) in a DHCP Discover message if the prefix of the Vendor Class (DHCP Option 60) or the prefix of the Client Hardware Address (chaddr field in DHCP Message) has certain values. Although this example is simple, the logic used to process a DHCP message can be more extensive.

LDRM can perform other functions as well. For example, LDRM can provide a suggested subnet to the DHCP Service from which it selects an IP address to lease to a client. Another example is executing authorization-related logic, and instructing the DHCP Service to drop a client packet and not respond to it.

□

2 Installation

Overview

Purpose

This chapter contains the prerequisites and instructions for installing and uninstalling LDRM.

Contents

This information presents the following topics.

□

Hardware prerequisites

The following amounts of disk space are required for each LDRM component:

- LDRM: 60 MB
- Jython: 10 MB (Enterprise only)
- Java: 85 MB (Enterprise only)

□

Software prerequisites

The following are software prerequisites for LDRM:

- VitalQIP 6.2 and all its prerequisites. For information on these requirements, see the *VitalQIP Installation Guide*.
- Lucent DHCP 5.4 build 22 or higher
- Java Development Kit J2SE 1.4.x
- Jython 2.1

□

Installation instructions

Install Java

Because rules compile only on the enterprise server, you only have to install Java on the enterprise server.

> **Important!** You must know the pathname where Java is installed before installing LDRM.

If you need to install the Java Development Kit, you can obtain it for each platform as follows:

Solaris, Linux, Windows	Sun's Java Technology pages
AIX	IBM's DeveloperWorks Java Technology pages
HP	HP's Java Technology pages

Install Jython

Because rules compile only on the enterprise server, you only have to install Jython on the enterprise server.

> **Important!** You must know the pathname where Jython is installed before installing LDRM.

You can obtain Jython by going to Jython's website and selecting the option to download Jython 2.1. The installation command for all operating systems except AIX is as follows:

```
java jython-21 -o /opt/jython21
```

> **Important!** When you install Jython on AIX, you must enter the `-f unix` option at the end of the command. For example:

```
java jython-21 -o /opt/jython21 -f unix
```

□

Install LDRM

When to use

This section describes how to install LDRM on your enterprise and remote servers.

Important! If you use a DHCP Failover Server, consider installing LDRM on it.

Before you begin

The screens shown in this chapter are for a Windows install. The install screens for other platforms have the same content, but a slightly different graphic appearance.

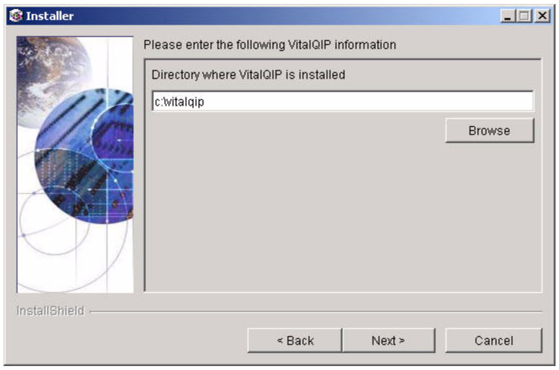
Procedure

1 Execute the installation file on your enterprise server(s) and remote server(s).

Platform	Installation file
Windows	*ldrm20setupwin32.exe*
AIX	*ldrm20SetupAix.bin*
HP	*ldrm20SetupHPux.bin*
Linux	*ldrm20SetupLinux.bin.*
Solaris	*ldrm20SetupSolaris.bin*

The welcome screen displays the product name, Lucent's name, and Lucent's website. Click **Next** to continue. The License Agreement screen displays.

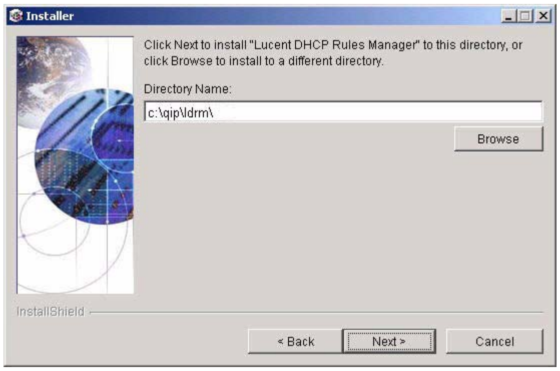
2 Read the license agreement. You must accept the license agreement to continue the installation. To continue, click **Accept** and then click **Next**. The VitalQIP information screen displays.


Please enter the following VitalQIP information

Directory where VitalQIP is installed

c:\vitalqip

3 Verify that the directory is the location of VitalQIP on your system. Change it if necessary, and click **Next** to continue. The Lucent DHCP Rules Manager Directory Name screen displays.

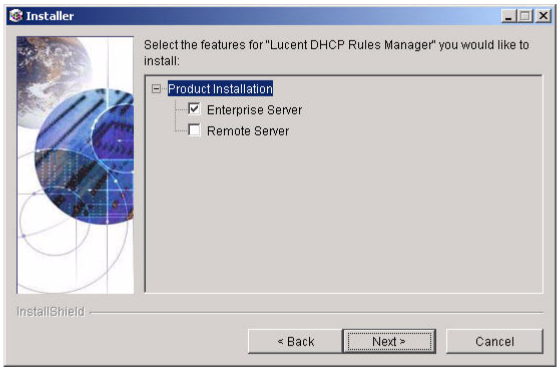


Click Next to install "Lucent DHCP Rules Manager" to this directory, or click Browse to install to a different directory.

Directory Name:

c:\qip\ldrm\

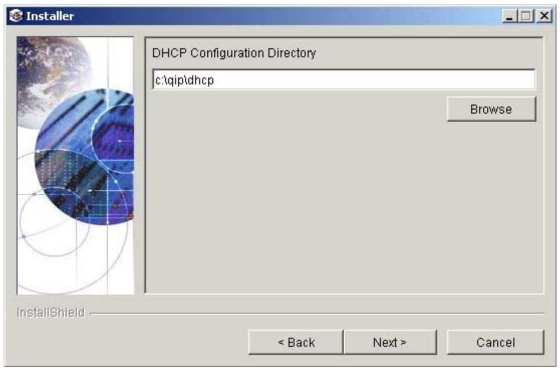
4 Verify the directory name for the LDRM directory is acceptable (Lucent recommends using the default value). The Product Features screen displays.



Select the features for "Lucent DHCP Rules Manager" you would like to install:

- Product Installation
  - [✓] Enterprise Server
  - [ ] Remote Server

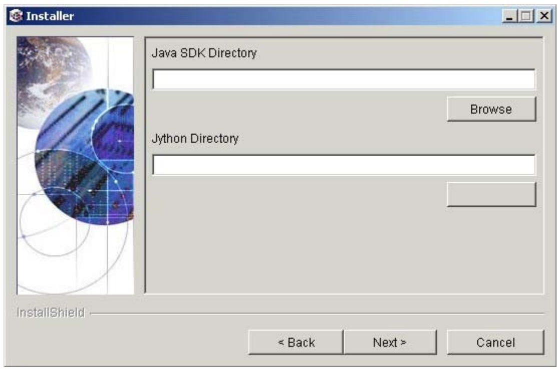
5 Select the Enterprise Server feature and/or the Remote Server feature. You can independently select and install both features, and you can install them both on the same system if you want. Click **Next** to continue. The DHCP Configuration Directory screen displays.


Installer

DHCP Configuration Directory

c:\qip\dhcp

6 Type the path to the DHCP configuration directory, The DHCP configuration directory defaults to the directory where DHCP is installed. Click **Next** to continue. The Java SDK/Jython Directories screen displays.

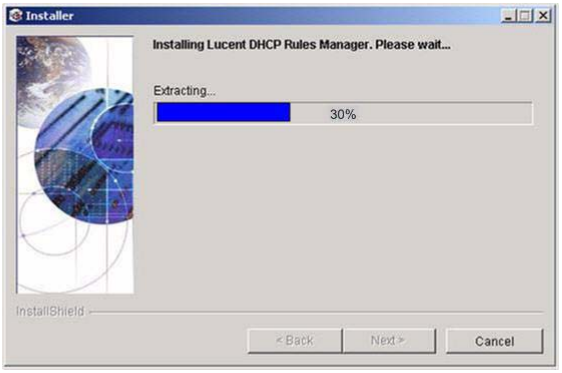

Installer

Java SDK Directory

Jython Directory

7 Type the path to the Java/SDK directory and the Jython directory. There are no defaults for these directories. Click **Next** to continue. The Install Preview screen displays a list of the selected features and the amount of disk space required.

8 Click **Next** to continue. The Installation Progress panel displays.


Installer

Installing Lucent DHCP Rules Manager. Please wait...

Extracting...

30%

This panel displays the current file being installed and a progress bar as the product is installed. When the installation is complete, click **Next** to continue. The Install Summary Panel displays a message stating if the install was a success or a failure.

END OF STEPS

Uninstall LDRM

When to use

This section describes how to uninstall LDRM.

Procedure

1 Ensure DHCP's LDRM callouts are turned off. See “Configure the DHCP Service for LDRM”, on page 20 for more information.

Important! You do not have to remove DHCP patches to uninstall LDRM.

2 Stop LDRM. See “Start and stop LDRM”, on page 27 for more information.

3 Execute the uninstall file for your platform. The uninstall file is located in the *_uninst* directory of the LDRM application.

Platform	Installation file
Windows	*uninstaller.exe* Execute from the Windows Control Panel using the Add/Remove Programs option.
AIX HP Linux Solaris	*uninstaller.bin*

4 For Windows, remove LDRM from the VitalQIP Service Controller.

For detailed information on the VitalQIP Service Controller, refer to the *VitalQIP Administrator Reference Manual*, Chapter 13, Manage VitalQIP Services (in VitalQIP Release 6.2, information on the VitalQIP Service Controller is located in Chapter 3).

END OF STEPS

☐

LDRM directory structure

The following shows the directory structure of LDRM after it is installed.

```
$QIPHOME/                                 VitalQIP Home Directory
          dhcp/                           DHCP configuration files
          ldrm/                           Example LDRM install location
                    bin/                  LDRM tools
                    conf/                 LDRM’s configuration and rules
                    stage/                Rule distribution (Enterprise)
                    userexits/            Example LDRM User Exits
                    _uninst/              Uninstaller
                    _jvm/                 Files used to uninstall LDRM
                    lib/                  Application files
                    etc/                  Files for the LDRM environment
```

□

3 Configuration

Overview

Purpose

This chapter contains information you need to configure LDRM, VitalQIP and the DHCP Server to utilize LDRM.

Contents

This information presents the following topics.

□

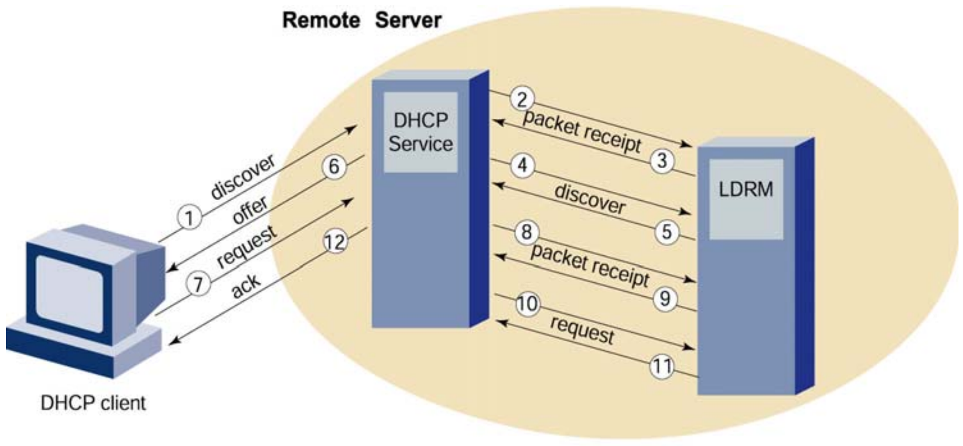
Configure the DHCP Service for LDRM

When to use

The interface between the Lucent DHCP Service and the LDRM Service is enabled/disabled by a set of DHCP server level policies. As the DHCP server interacts with clients, it uses any active LDRM callouts to check client input packets and process rules.

Figure 3 shows how the DHCP service interacts with LDRM.

Figure 3 LDRM/DHCP callout processing



LDRM/DHCP processing occurs as follows:

1. Discover from DHCP client to DHCP service

This discover packet becomes the packet receipt/discover LDRM pair.

2. Packet receipt from DHCP service to LDRM
3. Packet receipt from LDRM to DHCP service
4. Discover from DHCP service to LDRM
5. Discover from LDRM to DHCP service

After the DHCP Service processes the discover packet an offer packet is sent to the client.

6. Offer from DHCP service to DHCP client

7. Request from DHCP client to DHCP service

This request packet becomes the packet receipt/request LDRM pair.

8. Packet receipt from DHCP service to LDRM
9. Packet receipt from LDRM to DHCP service
10. Request from DHCP service to LDRM
11. Request from LDRM to DHCP service

After the DHCP Service processes the request packet the client is sent an ack packet.

12. Ack from DHCP service to DHCP client

Procedure

To set the policies that enable the callouts using the VitalQIP GUI, perform the following steps:

1 Select **Infrastructure/Server...**

2 On the Server Profile Option popup window, select the appropriate server and click **OK**.

3 On the Server Profile: Modify popup window, scroll down in the Parameters/Values list and select **Additional Policies**.

Important! For VitalQIP 6.2, the LDRM callout policies displayed on the GUI are non-functional. Be sure to follow the instructions here and do not use **Enable LDRM Interface** displayed elsewhere in the server profile. This issue will be addressed in a future release.

4 In the Value area, enter the following policy: `UseLDRMCallout=1`. You must set this policy to 1 to cause the DHCP server to load and initialize the LDRM callout library.

CAUTION

Service disruption hazard

If `UseLDRMCallout=1`, *then the LDRM Service must be running. If it is not, the DHCP Service does not respond to client requests.*

5 Also in the Additional Policies area, enable the individual API callouts in the file *DHCPD.pcy* that are appropriate for your LDRM rule file, as shown in Table 3, for example, `PacketReceiptLDRMCallout=1`.

Important! Each callout requires a certain amount of system resources. Only enable the ones you plan to use.

Table 3 LDRM API callouts

Callout	Values
PacketReceiptLDRMCallout	0 - (default) packet receipt callout is disabled 1 - packet receipt callout is enabled Only used to change hardware address and/or hostname after a packet has been received, but before it has been processed.
DiscoverLDRMCallout	0 - (default) discover callout is disabled 1 - discover callout is enabled
RequestLDRMCallout	0 - (default) request callout is disabled 1 - request callout is enabled
BootpRequestLDRMCallout	0 - (default) bootp request callout is disabled 1 - bootp request callout is enabled

END OF STEPS

LDRMCallout.pcy file

The configuration of the DHCP server's LDRM Callout Library is defined in a policy file named LDRM*Callout.pcy*. This file resides in *$QIPHOME/dhcp* and is installed during the LDRM installation with default values. If you want to change the values from their default, edit this file. The DHCP server must be restarted for new policy values to be recognized. The tags and values are described in Table 4:

Table 4 LDRMCallout.pcy file

Policy	Values
LDRMServerAddress	***Do not modify.*** IP address of server where LDRM Service is located. (default =127.0.0.1)
ReceiveTimeout	Numeric - timeout, in seconds, for socket connection with LDRM. (default = 5 seconds)
LDRMPortNumber	***Do not modify.*** Port number that is used for socket communications with LDRM. (default = 5678)
StatisticsDumpFrequency	numeric - 0 turns off callout statistics generation. A non-zero number indicates the number of LDRM responses that occur between generation of statistics. Statistics are written to the *$QIPHOME/dhcp/*LDRM*Callout.stats* file. (default = 0). For an example of this file, see “DHCP LDRM Callout Statistics”, on page 79
Debug	numeric - 0=minimal logging, 1=event logging, 2=full logging. (default = 1)
MaxDebugFileSize	numeric - size at which LDRM*Callout.log* is reinitialized. (default = 1,000,000)

☐

LDRM properties file

The LDRM service has its own property file, *ldrm.properties*, that defines a runtime environment. If you want to change the values from their default, edit this file. Its path is */opt/qip/ldrm/conf/ldrm.properties*. It contains the values shown in Table 5.

Table 5 LDRM.properties file

Property	Value
ldrm.port	***Do not modify***. LDRM server's listen port. (default=5678)
ldrm.admin.listen	LDRM's admin listen address. 0.0.0.0 accepts any connection. The recommended address is 127.0.0.1.
ldrm.admin.port	LDRM admin listen port. Use '`0`' to disable the feature.
ldrm.options	***Do not modify***. Option configuration file. (default=*optionValidations.txt*)
ldrm.ruleFile	***Do not modify***. Name of the *.jar* file. (default=*/opt/qip/ldrm/conf/rules.jar*)
Log4JConfigFile	***Do not modify***. LDRM's log4j configuration file. (default=*/opt/qip/ldrm/conf/ldrm-log4j.properties*)

The only properties you can change are `ldrm.admin.listen` and `ldrm.admin.port`. These properties configure an administrative web page that provides performance statistics. If you enable this feature, the URL is:

http://<SystemName>:<portNumber>

Where `SystemName` (or IP address) is the system running LDRM, and the `portNumber` is the value specified in `ldrm.admin.port`. See "LDRM Status", on page 76 for more information.

☐

Configure VitalQIP User Exits for LDRM

User Exits are a VitalQIP mechanism that enable adding customizations during File Generation. DHCP User Exits typically allow you to manipulate the DHCP configuration file after it has been created. Specifically, the user exit is called after the *dhcpd.pcy* and *dhcpd.conf* files are created, but before the DHCP service is notified to refresh the configuration file. The VitalQIP Remote Service saves all received configuration files to disk on the remote server and then calls the appropriate user exit for the platform, release and server:

Server	**Windows**	**Unix**
Remote Server	*qipdhcpuserexit.bat*	*qipdhcpuserexit*
Enterprise Server	*qipdhcpuserexitfgs.bat*	*qipdhcpuserexitfgs*

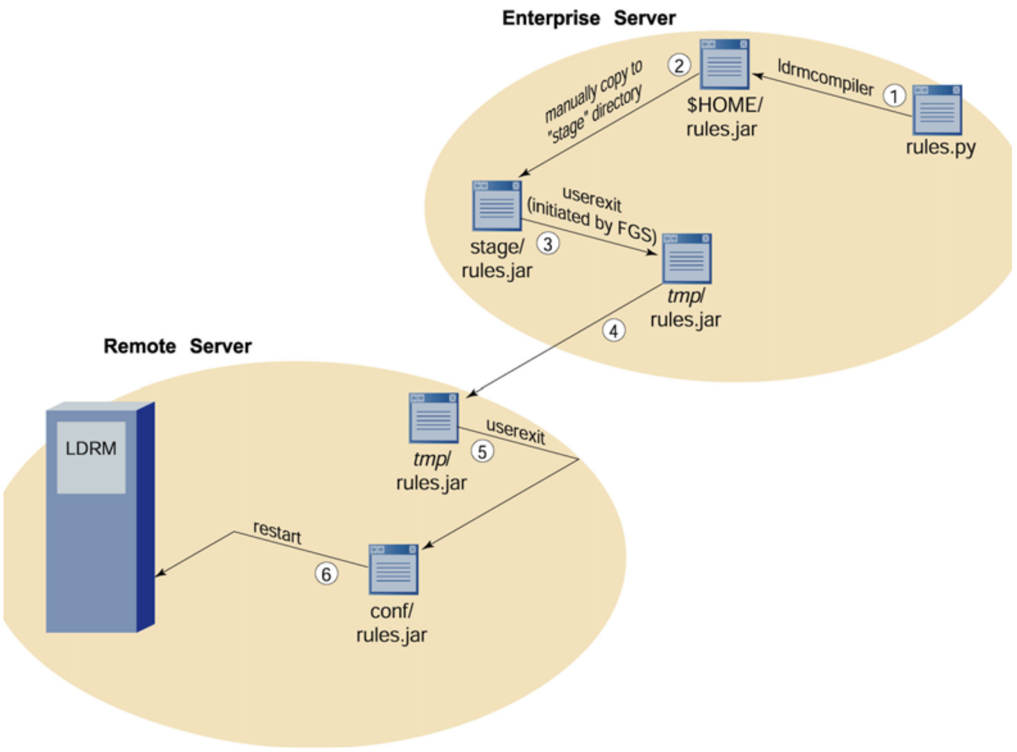
If a DHCP user exit exists, you must manually integrate the existing user exit with LDRM's. LDRM provides a reference copy of its user exits in *<ldrmhome>/userexits* directory. LDRM uses this customization mechanism first as a means to transport the rules file from the enterprise server to the selected remote servers, and second as a means to restart the remote LDRM service. You can use the provided samples. Be sure your scripts are executable.

If a DHCP user exit does not exist, then LDRM's installer automatically copies it into *$QIPHOME/userexits*. As shown above, there is one user exit for a remote server and another for the enterprise server.

Important! When uninstalling, LDRM's installer does not remove a user exit unless it was previously installed by LDRM and the file was not modified after it was installed. To ensure proper VitalQIP operation, be sure to manually check the user exits after an uninstall.

As shown in Figure 4, LDRM uses the user exit mechanism to obtain a copy of *rules.jar*, if any, from *<ldrmhome>/stage*. The VitalQIP File Generation Service (invoked from the VitalQIP GUI) then transfers *rules.jar*, along with the standard DHCP files, to the selected remotes. On each remote server, LDRM's *qipcnfuserexit* installs *rules.jar* under LDRM's *conf* directory, and then restarts LDRM.

Figure 4 LDRM user exit process



Start and stop LDRM

When to use

After you install and configure LDRM, you must start it on the remote servers. Instructions for Windows and Unix platforms appear in the following sections.

Start LDRM in a Windows environment

During the Windows installation, the installation program creates a service called Lucent DHCP Rules Manager. After the installation is complete, you must add this service to the VitalQIP Service Controller.

Procedure

Use the following steps:

1 Access **Start|Programs| VitalQIP| Service Controller**. The Service Controller window opens.

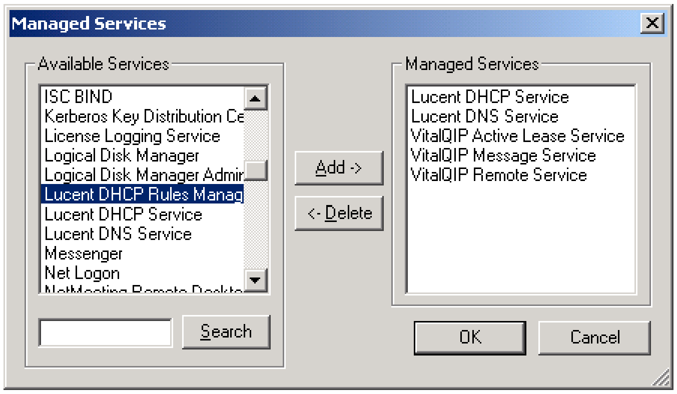
Important! Step 2 to step 8 are only required when LDRM is initially installed. If you have previously performed these steps, continue to step 9 to start LDRM.

2 In the Service Controller window, click **Configure.** The Configure Services window opens.

3 Click **Services...**

4 Type Lucent DHCP Rules Manager and click **Search**.

5 Highlight Lucent DHCP Rules Manager and click **Add**. Then click **OK**.



6 In the Configure Service Controller window, select **Lucent DHCP Rules Manager** from the Managed Services list.

7 In the Startup Type box, select **Automatic**. This causes LDRM to start when Windows boots.

8 Click **OK**. The Service Controller window is displayed.

9 In the Service Controller window, select **Lucent DHCP Rules Manager**. Click **Start** to start LDRM.

10 To stop LDRM, Select **Lucent DHCP Rules Manager** in the Service Controller window and click **Stop**.

For detailed information on the VitalQIP Service Controller, see the *VitalQIP Administrator Reference Manual*, Chapter 13, Manage VitalQIP Services (in VitalQIP Release 6.2, information on the VitalQIP Service Controller is located in Chapter 3).

Start LDRM in an AIX, HP, Solaris or Linux environment

During the AIX, HP, Solaris or Linux installation, two scripts are created to start and stop LDRM. You must run the start script when you start the remote server where LDRM is installed. The script names are:

ldrmstart.sh	Start LDRM
ldrmstop.sh	Stop LDRM

END OF STEPS

□

4 Rules

Overview

Purpose

This chapter describes how you create and test LDRM rules. Rules allow you to modify DHCP field and option values of both incoming and outgoing DHCP packets.

Contents

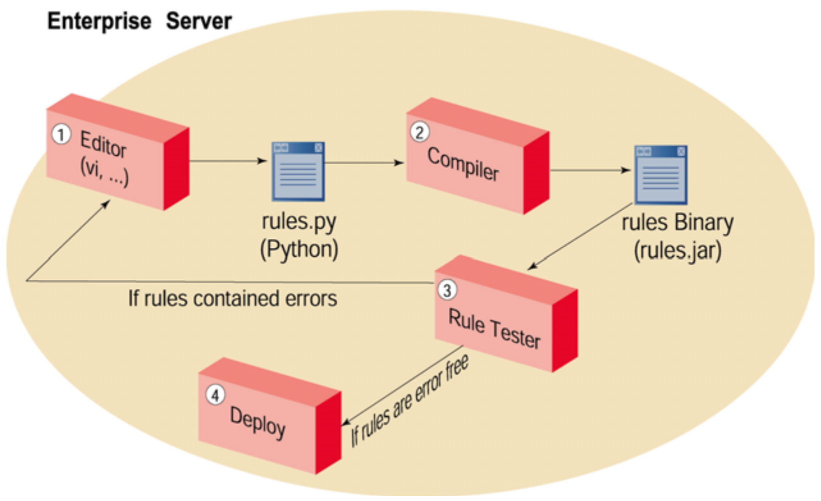
This information presents the following topics.

Rules overview

Rules allow you to modify DHCP field and option values of both incoming and outgoing DHCP packets. They are also capable of specifying whether the packet is authorized for continued DHCP service processing. For example, LDRM allows you to assign a user class to a client based on the client's MAC address. You create rules using the Python programming language.

The rules are executed in a rules engine, which uses the input DHCP packet, user-defined local variables and LDRM-provided library methods for logic execution. After you create a rule, you test it with the LDRM rule tester. Figure 5 shows the process you follow to create and test rules.

Figure 5 Rules creation process



In a production environment, you only modify output variables. However, when you test rules before you deploy them, you must run them against an input packet file. The packet file contains values that are found in a DHCP request packet in a production environment. To create a packet file, you must assign values to input variables.

> **Important!** Creating an input packet file for testing is the only time you assign values to input variables.

The following sections describe creating and testing rules, and variable programming.

Create rules

When to use

You create rules on the enterprise server only. When you create rules, you can check the incoming DHCP packet values, specified in Table 7, “Input variables”, on page 35. Rules can set the values specified as output variables in Table 8, “Output Variables used with Packet Receipt API callouts.”, on page 38 and Table 9, “Output variables used with Discover, Request and Bootp Request API callouts”, on page 39. The output variables are returned to the DHCP server to affect subsequent packet processing or to directly set values in the packet that DHCP returns to the client.

Procedure

Follow these steps when creating rules.

1 On your enterprise server, use a text editor to create a file called *rules.py*. This is the file where you type your rules, using standard Jython programming commands. The following is an example of some rules syntax:

```
if dhcpIn.getChAddr() == "01:02:03:04:05:06" :
   dhcpOut.setOption("user-class", "student")
   dhcpOut.setOption("lease-time", 3600)
```

For an example of a *rules.py* file, see “Example rules file”, on page 52.

Lucent recommends not creating new rules in the *ldrm/conf/* directory in a production environment. Create and test rules in another directory, and only move them to the stage directory after you have verified them.

You can organize rules hierarchically in multiple levels. You can code rules in separate files, or in one large file. If you use multiple files, the first one must be named *rules.py*. It calls any additional files, to which you can assign any name, provided it has a suffix of .py. Because LDRM imposes a single .jar file limitation, all the rules are compiled into a single *rules.jar* file. See “Vendor specific information”, on page 49 for an example of a *rules.py* file that calls additional files.

2 Compile the *rules.py* file into a *rules.jar* file by running one of the following from a command line

```
ldrmcompiler.sh rules.py          (Unix Environment)
ldrmcompiler.bat rules.py         (Windows Environment)
```

This creates a *rules.jar* file in the same directory. It also creates a *jpywork* subdirectory in this directory. If the *jpywork* directory is deleted, it is re-created next time you compile a *rules.jar* file.

Important! The compiler only checks for syntax errors. It is only at runtime that semantic (logic) errors are found. For example, misspelling a variable, or perhaps an option's name, is not detected by the compiler. This is why testing rules before deployment is strongly recommended.

Lucent recommends you test the *rules.jar* file using *ldrmtester.sh*. Refer to "Test LDRM rules", on page 54 for instructions.

3 Using standard operating system commands, copy the *rules.jar* file to *$LDRMHOME/stage*.

4 Use the VitalQIP File Generation Utility to distribute the *rules.jar* file to the other servers on the network. The VitalQIP File Generation Service pushes the rules to the *$LDRMHOME//conf* directory on the remote servers. See the DHCP Generation section in Chapter 8 of the *VitalQIP Adminstrator Reference Manual* for more information (in VitalQIP Release 6.2, this information is located in Chapter 5 of the *VitalQIP User's Guide*).

END OF STEPS

Rules syntax

The following is a summary of most of the syntax you need to write rules.

Indentation

The line's indentation level, not the use of curly braces, is used to group statements together.

Datatype checking

Datatypes are checked at runtime, not compile time. For example, you do not declare a variable, such as `integer i` before use.

Conditional processing

Conditional statements use the keywords: "`if`", "`elif`" and "`else`". You must place a colon at the end of the statement.

Comparing values

Comparison operators are: "==", "`!=`", ">", "<".

Boolean logic

- Boolean operators are: "`not`", "`and`", "`or`". You can use parentheses.
- Boolean values are 1 (true), or 0 (false). For convenience LDRM defines two variables named "on" and "off", which are respectively set to 1 and 0.

Series of values

A list of values is enumerated inside square brackets: [item1, item2, …]. For example

```
dhcpOut.setOption( "routers", ["10.20.30.40","50.60.70.80"])
```

Lists may be used in many places, including condition statements, or where appropriate, as a DHCP parameter value.

```
if myvar in [<setoption1, setoption2 . . .>]:
```

Variables

When using your own variables, you must assign a value before using them. Assignment statements are supported as:

```
yourVariable = someValue
```

Important! A common coding error is to misspell variables. Misspelled variables are not detected by the compiler.

Readability

- To improve readability, you can split long lines with a backslash.
- Comments start with a #

Library methods

An assortment of library methods are available, either provided by LDRM, or through the Jython language. For more information on the Jython language, see ***www.jython.org***.

Exit actions

Exit actions for rule definitions are handled with the “return” language statement.

Jython documentation

For more information on the Jython language, see ***www.jython.org*** and select the *Documentation* link.

List of data types

Refer to Table 6 for a list of the data types used when creating a rules file:

Table 6 Data types and examples

Data type	Example
String	Any combination of alphanumeric characters.
Binary String	A list of bytes stored as a string
IP Address	A standard dotted decimal notation. For example: `155.108.25.12`
IP and Mask	String: “192.168.77.1/255.255.255.0”
MAC Address	A string with colons between bytes. For example: 00:06:5B:63:09:5D
Integer	Any number that does not contain a decimal point. Do not use commas when entering numbers over 999.

□

Rule variables

Using variables

Rules files contain both input and output variables. Input variables are the values that a rule can read from an incoming DHCP packet. Output variables are the values a rule can override in an outgoing DHCP packet.

Input variables

In LDRM, the following variable fields can be read from incoming DHCP packets. These are the values read and processed by LDRM from DHCP packets.

Important! Never assign values to these variables in a rules file. These variables are read only values. LDRM rules can read these variables, and implement a rule based on them, but they cannot be changed. In a production environment, these variables come from the client that is requesting a DHCP lease. Refer to “Example rules file”, on page 52 for a sample rules file.

Table 7 Input variables

Variable	Description	Usage
BootFile	Field in DHCP Message. String specifying fully qualified directory pathname of the bootfile.	Invoked with `dhcpIn.getBootFile()`
ChAddr	Field in DHCP Message. Client hardware address. Alphanumeric string.	Invoked with `dhcpIn.getChAddr()`
CiAddr	Field in DHCP Message. Client IP address; only filled in if client is in BOUND, RENEW or REBINDING state and can respond to ARP requests.	Invoked with `dhcpIn.getCiAddr()`

DhcpEvent	Specifies the callout the DHCP server used to send a message to LDRM. See Table 3, “LDRM API callouts”, on page 22 for additional information. Values are:
	`packetreceipt`
	`discover`
	`request`
	`bootprequest`
		Invoked with `dhcpIn.getDhcpEvent()`

Variable	**Description**	**Usage**
Flags	Field in DHCP Message. To work around some clients that cannot accept IP unicast datagrams, DHCP uses the “flags” field. The leftmost bit is defined as the BROADCAST (B) flag. The remaining bits of the flags field are reserved for future use.	Invoked with `dhcpIn.getFlags()`
GiAddr	Field in DHCP Message. Relay agent IP address.	Invoked with `dhcpIn.getGiAddr()`
HLen	Field in DHCP Message. Hardware address length.	Invoked with `dhcpIn.getHLen()`
Hops	Field in DHCP Message. Client sets to zero, optionally used by relay agents when booting via a relay agent.	Invoked with `dhcpIn.getHops()`
HType	Field in DHCP Message. Hardware address type.	Invoked with `dhcpIn.getHType()`
Op	Field in DHCP Message. 1 =Request to DHCP server 2 = Reply from DHCP server	Invoked with `dhcpIn.getOp()`
Secs	Field in DHCP Message. Filled in by client, seconds elapsed since client began address acquisition or renewal process.	Invoked with `dhcpIn.getSecs()`
TftpServer	Field in DHCP Message. String. Specifies the IP address of the TFTP server.	Invoked with `dhcpIn.getTftpServer()`
XID	Field in DHCP Message. Transaction ID, a random number chosen by the client, used by the client and server to associate messages and responses between a client and a server.	Invoked with `dhcpIn.getXID()`
vendor-specific	DHCP Option 43. Alphanumeric string Not used for Bootp. May contain encapsulated sub-options. See “Vendor specific information”, on page 49 for more information.	Invoked with `dhcpIn.getOption(“vendor_specific”)`
requested-ip-addr	DHCP Option 50. IP Address.	Invoked with `dhcpIn.getOption(“requested-ip-addr”)`

Variable	Description	Usage
dhcp-lease-time	DHCP Option 51. Integer. Indicates lease time in seconds. A value of `-1` indicates an indefinite lease.	Invoked with `dhcpIn.getOption("dhcp-lease-time")`
dhcp-message-type	DHCP Option 53. Integer.	Invoked with `dhcpIn.getOption("dhcp-message-type")`
dhcp-server-id	DHCP Option 54. IP Address.	Invoked with `dhcpIn.getOption("dhcp-server-id")`
parameter-request-list	DHCP Option 55. Binary string.	Invoked with `dhcpIn.getOption("parameter-request-list")`
vendor-class	DHCP Option 60. Text.	Invoked with `dhcpIn.getOption("vendor-class")` Not used for Bootp
dhcp-client-id	DHCP Option 61. Text.	Invoked with `dhcpIn.getOption("dhcp-client-id")`
user-class	DHCP Option 77. Text. Not used for Bootp.	Invoked with `dhcpIn.getOption("user-class")`
client-FQDN	DHCP Option 81. String whose length is between 3 and 255. Not used for Bootp. See "LDRM rule library", on page 64 for more information.	Invoked with `dhcpIn.getOption("client-FQDN")`
relay-agent-info	DHCP Option 82. List of binary strings, each list member contains a suboption. The first list member is numbered [0].	Invoked with `dhcpIn.getOption("relay-agent-info")`
subnet-selection	DHCP Option 118. IP Address.	Invoked with `dhcpIn.getOption("subnet-selection")`

Output variables

You can use the variables in this section in your rules file. LDRM uses the variables in this section to override values in the outgoing DHCP packet sent to clients.

For each `dhcpOut.set` command, note that strings must be in quotes, and numeric values must not be in quotes. Syntax is shown in the tables below.

Packet Receipt API callouts

For Packet Receipt API callouts, the LDRM rules support the assignment of values to the DHCP variables in Table 8. If validation fails for a Packet Receipt API callout, then `setAuthorized` is set to `off` and a Return Value of 0 is sent to the DHCP Server. Also, the specified validations are performed on variable values set in the LDRM rules.

Important! If validation fails, the following occurs:

- LDRM reports errors to LDRM.*log*
- LDRM automatically turns off authorization
- the DHCP server discards the packet and writes an entry to the DHCP server log (*dhcpd.log*)

Table 8 Output Variables used with Packet Receipt API callouts.

Variable	Description	Validation	Usage
host-name	DHCP Option 12	String. Length is between 1 and 63 bytes, inclusive.	Invoked with `dhcpOut.setOption("host-name", "`*`<host name>`*`")`
chaddr	DHCP Server parameter	MAC Address.	Invoked with `dhcpOut.setChAddr("`*`<MAC Address>`*`")`

Discover, Request and Bootp Request API callouts

For Discover, Request, and Bootp Request API callouts, the LDRM rules support the assignment of values to the DHCP variables in the following table. (There are some exceptions in which some variables are not used in Bootp Request API callouts, and these are identified in the rows below.) Additionally, the specified validations are performed on variable values set in the LDRM rules. (Each DHCP Option in Table 9 is described in the RFC specified in the first table at ***http://www.iana.org/assignments/bootp-dhcp-parameters***.) If validation fails, then `setAuthorized` is set to `off` and a Return Value of 0 is sent to the DHCP Server.

Important! If validation fails, the following occurs:

- LDRM reports errors to LDRM.log
- LDRM automatically turns off authorization
- the DHCP server discards the packet and writes an entry to the DHCP server log (*dhcpd.log*)

Table 9 Output variables used with Discover, Request and Bootp Request API callouts

Variable	Description	Validation	Usage
Authorize	DHCP Server Parameter	Value is 0 (False) or 1 (True).	Invoked with `dhcpOut.setAuthorized(<0 or 1>)` If in an LDRM rule, a value is assigned to a DHCP variable, and it fails its validation, then the Authorize parameter is set to False for Discover, Request, and Bootp Request API callouts.
Boot File	Field in DHCP Message (file field in Section 2 of RFC 2131)	String. Length is between 1 and 128 bytes, inclusive.	Invoked with `dhcpOut.setBootFile("<filename>")` Sent to client when set by LDRM.
Suggested Subnet	DHCP Server Parameter	Formatted in standard dotted-decimal notation. (that is, ###.###.###.###).	Invoked with `dhcpOut.setSuggestedSubnet("<IP Address>")` Set by LDRM, and used by server to select the subnet from which the IP address is offered.
TFTP Server	Field in DHCP Message (siaddr field in Section 2 of RFC 2131)	Formatted in standard dotted-decimal notation (that is, ###.###.###.###).	Invoked with `dhcpOut.setTftpServer("<IP Address>")` Sent to client when set by LDRM.
time-offset	DHCP Option 2	Integer.	Invoked using `dhcpOut.setOption("time-offset", <number of seconds>)` Offset (in seconds) of the client's subnet from UTC. Signed integer
routers	DHCP Option 3	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("routers", ["<IP Address1>", "<IP Address 2>"...])` List of IP addresses
time-servers	DHCP Option 4	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("time-servers", ["<IP Address 1>", "<IP Address 2>" ...])`

Variable	Description	Validation	Usage
name-servers	DHCP Option 5	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("name-servers", ["`*`<IP Address 1>`*`", "`*`<IP Address 2>`*`" ...])`
domain-name-servers	DHCP Option 6	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("domain-name-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
log-servers	DHCP Option 7	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("log-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
cookie-servers	DHCP Option 8	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("cookie-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
LPR- servers	DHCP Option 9	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("LPR-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
impress-servers	DHCP Option 10	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("impress-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
resource-location-servers	DHCP Option 11	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("resource-location-servers", ["<IP Address 1>", "<IP Address 2>" ...])`

Variable	**Description**	**Validation**	**Usage**
host-name	DHCP Option 12	String. Length is between 1 and 63 bytes, inclusive.	Invoked using `dhcpOut.setOption("host-name", "<hostname>")` Sent to client if changed by LDRM
boot-size	DHCP Option 13	Integer. Value is between 1 and 65,535, inclusive.	Invoked using `dhcpOut.setOption("boot-size", <file length>)` File length in 512-octet blocks.
merit-dump	DHCP Option 14	String. Length is between 1 and 255 bytes, inclusive.	Invoked using `dhcpOut.setOption("merit-dump", "<ASCII string>")` NVT ASCII character string
domain-name	DHCP Option 15	String. Length is between 1 and 190 bytes, inclusive.	Invoked using `dhcpOut.setOption("domain-name", "<domain name>")` Sent to client if changed by LDRM
swap-server	DHCP Option 16	IP Address. Formatted in standard dotted-decimal notation.	Invoked using `dhcpOut.setOption("swap-server", "<IP Address>")`
root-path	DHCP Option 17	String. Length is between 1 and 255 bytes, inclusive.	Invoked using `dhcpOut.setOption("root-path", "<ASCII string>")` NVT ASCII character string
extensions-path	DHCP Option 18	String. Length is between 1 and 255 bytes, inclusive.	Invoked using `dhcpOut.setOption("extensions-path", "<ASCII string>")` NVT ASCII character string
IP-forwarding	DHCP Option 19	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption("IP-forwarding", <0 or 1>)`
non-local-source-routing	DHCP Option 20	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption("non-local-source-routing", <0 or 1>)`

Variable	Description	Validation	Usage
policy-filter	DHCP Option 21	IP addresses and associated masks are formatted in standard dotted-decimal notation, and must occur in pairs. Maximum length is 31 IP addresses and mask pairs.	Invoked using `dhcpOut.setOption("policy-filter", ["<IP Address>/<mask pair>", "<IP Address>/<mask pair>" . . .])` List of IP address and mask pairs For example `["192.168.100.1/255.255.255.0", "172.16.100.2/255.255.0.0"]`
max-dgram-reassembly	DHCP Option 22	Integer. Value is between 576 and 65,535, inclusive.	Invoked using `dhcpOut.setOption("max-dgram-reassmbly", <maximum datagram length>)` Maximum length of datagram
default-IP-TTL	DHCP Option 23	Integer Value is between 1 and 255, inclusive.	Invoked using `dhcpOut.setOption("default-IP-TTL", <number of seconds>)` Time To Live in seconds
path-MTU-aging-timeout	DHCP Option 24	Integer. Value is between 0 and 2,147,483,647, inclusive.	Invoked using `dhcpOut.setOption("path-MTU-aging-timeout", <number of seconds>)` Timeout in seconds
all-subnets-local	DHCP Option 27	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption("all-subnets-local", <0 or 1>)`
broadcast-address	DHCP Option 28	IP Address. Formatted in standard dotted-decimal notation.	Invoked using `dhcpOut.setOption("broadcast-address", "<IP Address>")`
perform-mask-discovery	DHCP Option 29	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption("perform-mask-discovery", <0 or 1>)`
mask-supplier	DHCP Option 30	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption("mask-supplier", <0 or 1>)`
router-discovery	DHCP Option 31	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption("router-discovery", <0 or 1>)`

Variable	Description	Validation	Usage
router-solicitation-address	DHCP Option 32	IP Address. Formatted in standard dotted-decimal notation.	Invoked using `dhcpOut.setOption("router-solicitation-address", "<IP Address>")`
static-routes	DHCP Option 33	IP addresses are formatted in standard dotted-decimal notation, and must occur in pairs. Maximum length is 31 pairs of IP addresses.	Invoked using `dhcpOut.setOption("static-routes", ["<IP Address/IP Address>", "<IP Address/IP Address>" . . .])`
trailer-encapsulation	DHCP Option 34	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption("trailer-encapsulation", <0 or 1>)`
arp-cache timeout	DHCP Option 35	Integer. Value is between 0 and 2,147,483,647, inclusive.	Invoked using `dhcpOut.setOption("arp-cache-timeout", <number of seconds>)`
ieee802-3 encapsulation	DHCP Option 36	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption("ieee802-3-encapsulation", <0 or 1>)`
default-tcp-ttl	DHCP Option 37	Integer. Value is between 1 and 255, inclusive.	Invoked using `dhcpOut.setOption("default-tcp-ttl", <number of seconds>)`
tcp-keepalive-interval	DHCP Option 38	Integer. Value is between 0 and 2,147,483,647, inclusive.	Invoked using `dhcpOut.setOption("tcp-keepalive-interval", <number of seconds>)`
tcp- keepalive garbage	DHCP Option 39	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption(tcp-keepalive-garbage", <0 or 1>)`
nis-domain	DHCP Option 40	String. Length is between 1 and 255 bytes, inclusive.	Invoked using `dhcpOut.setOption("nis-domain", "<domain name>")`
nis-servers	DHCP Option 41	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("nis-servers", ["<IP Address 1>", "<IP Address 2>" ...])`

Variable	Description	Validation	Usage
ntp-servers	DHCP Option 42	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("ntp-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
vendor-specific	DHCP Option 43	Length is between 1 and 255 bytes, inclusive. Prohibited in Bootp Request callouts. Set by client/LDRM, and sent to client when server policy EchoClientVendorInfo = 1. **Important!** The default value for the EchoClientVendorInfo policy in the DHCP server profile is 0. This policy must be set in the Additional Policies section of the server profile. Do not enable this policy if you are supporting PXE clients. See "Vendor specific information", on page 49 for more information.	Invoked using `dhcpOut.setOption("vendor-specific", "<value>")` .
netbios-name-servers	DHCP Option 44	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("netbios-name-servers", ["<IP Address 1>", "<IP Address 2>" ...])` List of IP addresses
netbios-dd-servers	DHCP Option 45	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("netbios-dd-servers", ["<IP Address 1>", "<IP Address 2>" ...])` List of IP addresses
netbios-node-type	DHCP Option 46	Valid values are 1, 2, 4, and 8.	Invoked using `dhcpOut.setOption("netbios-node-type", "<node type>")`

Variable	Description	Validation	Usage
font-servers	DHCP Option 48	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("font-servers", ["<IP Address 1>", "<IP Address 2>" ...])` List of IP addresses
x-display-manager	DHCP Option 49	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("x-display-manager", ["<IP Address 1>", "<IP Address 2>" ...])` List of IP addresses
requested-IP-addr	DHCP Option 50	IP Address. Formatted in standard dotted-decimal notation. Prohibited in Bootp Request callouts.	Invoked using `dhcpOut.setOption("requested-IP-addr", "<IP Address>")` Set by client/LDRM, and used by server to select IP address.
DHCP -lease-time	DHCP Option 51	Integer. Prohibited in Bootp Request callouts. A value of `-1` indicates an indefinite lease. Set by the client/LDRM, and used by the server as requested leasetime. This option is sent to client in offer/ack if HonorRequestedLease Time policy is enabled, or the requested value is less than what would be assigned from configuration data.	Invoked using `dhcpOut.setOption("DHCP-lease-time", <seconds in lease>)`
dhcp-renewal-time	DHCP Option 58	Integer. Value is between 0 and 2,147,483,647, inclusive. Prohibited in Bootp Request callouts.	Invoked using `dhcpOut.setOption("dhcp-renewal-time", <number of seconds>)`
dhcp-rebinding-time	DHCP Option 59	Integer. Value is between 0 and 2,147,483,647, inclusive. Prohibited in Bootp Request callouts.	Invoked using `dhcpOut.setOption"dhcp-rebinding-time", <number of seconds>)`

Variable	Description	Validation	Usage
vendor-class	DHCP Option 60	String. Prohibited in Bootp Request callouts.	Invoked using `dhcpOut.setOption("vendor-class", "<vendor class>")` Set by client/LDRM, and used by server to select IP address, set options, and control server behavior (for example, address shuffling). Not sent to client.
nis+ domain	DHCP Option 64	String. Length is between 1 and 255 bytes, inclusive.	Invoked using `dhcpOut.setOption("nis+-domain", "<domain string>")`
nis+ servers	DHCP Option 65	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("nis+-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
mobile-ip-home-agents	DHCP Option 68	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("mobile-ip-home-agents", ["<IP Address 1>", "<IP Address 2>" ...])`
smtp-servers	DHCP Option 69	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("smtp-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
pop3-servers	DHCP Option 70	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("pop3-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
nntp-servers	DHCP Option 71	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("nntp-servers", ["<IP Address 1>", "<IP Address 2>" ...])`

Variable	Description	Validation	Usage
www-servers	DHCP Option 72	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("www-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
finger-servers	DHCP Option 73	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("finger-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
irc-servers	DHCP Option 74	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("irc-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
streettalk-servers	DHCP Option 75	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("streettalk-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
dhcp-stda-servers	DHCP Option 76	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption("dhcp-stda-servers", ["<IP Address 1>", "<IP Address 2>" ...])`
user-class	DHCP Option 77	String. Prohibited in Bootp Request callouts.	Invoked using `dhcpOut.setOption("user-class", "<user_class>")` Only single-value User Class formats are supported, and multiple User Class values are not supported. Set by client/LDRM, and used by server to select IP address, set options, and control server behavior (for example, address shuffling). Not sent to client.

Variable	Description	Validation	Usage
client-FQDN	DHCP Option 81	Length is between 3 and 255 bytes, inclusive. Prohibited in Bootp Request callouts. See “LDRM rule library”, on page 64 for more information.	Invoked using `dhcpOut.setOption(“client-FQDN”, “<client FQDN>”)` Set by client/LDRM, and used by server to affect dynamic DNS processing.
novell-nds-servers	DHCP Option 85	Each IP address is formatted in standard dotted-decimal notation, and the maximum number of IP addresses is 63.	Invoked using `dhcpOut.setOption(“novell-nds-servers”, [“<IP Address 1>”, “<IP Address 2>” ...])`
ipv4-auto-configuration	DHCP Option 116	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption(“ipv4-auto-configuration”, <0 or 1>)`

□

Vendor specific information

This option is used by clients and servers to exchange vendor-specific information. When you retrieve this information with LDRM, you initially have a binary string. Jython provides various modules to manage binary data, including *struct*. For the following Microsoft-specific suboptions, LDRM provides a helper method for manipulating this data.

The following example shows how LDRM *MSFT* manipulates data.

```
# First obtain the raw data
rawData = dhcpIn.getOption('vendor-specific')
# To easily decode the raw binary data:
      msft = MSFT( rawData )    # This will format it.

      print 'Disable NetBIOS=', \
                msft.getDisableNetBIOS()
      print 'Release Lease On Shutdown=', \
                msft.getReleaseLeaseOnShutdown()
      print 'Get Default Router Metric Base=', \
              msft.getDefaultRouterMetricBase()
```

Table 10   Vendor-specific variables

Variable	Description	Validation	Usage
Disable NetBIOS over TCP/IP	Sub-option Code 1. Encapsulated in DHCP Option 43	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption(<0 or 1>)` This is a Microsoft vendor-specific code.*
Release DHCP Lease on Shutdown	Sub-option Code 2. Encapsulated in DHCP Option 43	Value is 0 (False) or 1 (True).	Invoked using `dhcpOut.setOption(<0 or 1>)` This is a Microsoft vendor-specific code.*
Default Router Metric Base	Sub-option Code 3. Encapsulated in DHCP Option 43	Value is between 1 and 9,999, inclusive.	Invoked using `dhcpOut.setOption(<#>)` This is a Microsoft vendor-specific code.*

* See the “Microsoft Options” subsection of “DHCP Tools and Settings” page for Windows Server 2003 at ***www.microsoft.com/resources/documentation/WindowsServ/2003/all/techref/en-us/Default.asp?url=/Resources/Documentation/windowsserv/2003/all/techref/en-us/w2k3tr_dhcp_tools.asp***

☐

Option 81 client FQDN

Option 81 is a suboption format option. It contains the data elements described in Table 11.

Table 11 Option 81 data elements

Element	Definition
byte 1	`flags byte` - low order bit is the “S” bit, when set by client or LDRM, requests that DHCP server perform the A resource record DNS update, in addition to the PTR resource record update that the server is always responsible for. When this bit is clear, the client or LDRM requests that the client perform the A resource record DNS update
byte 2	`RCODE1` - response code 1, set by server to 255 (hex ‘FF’) when the DHCPACK has been sent without waiting for PTR record update to complete. Otherwise 0.
byte 3	`RCODE2` - response code 2, set by server to 255 (hex ‘FF’) when the DHCPACK has been sent without waiting for A record update to complete. Otherwise 0.
bytes 4-n	ASCII. Fully qualified domain name.

The use of this option by the DHCP server is controlled by the Option81Support server policy value defined for the DHCP server. If rules are to include setting option 81 data, the value of `Option81Support` must be set to Client, Server, or Ignore. See “DHCP Server Policies” in Chapter 4 of the *VitalQIP Administrator Reference Manual* for definitions of these values (in VitalQIP Release 6.2, these definitions are located in Chapter 2 of the *VitalQIP User’s Guide*).

The following is used to handle option Option 81 Client FQDN.

```
rawData = dhcpIn.getOption(‘client-FQDN’)
if rawData is not None :
   fqdn = FQDN(rawData)
   print ‘S-bit=’, fqdn.getSbit()
   print ‘fqdn=’, fqdn.getFQDN()
```

The 'set' methods are as follows:

```
    fqdn.setSbit(boolean)
    fqdn.setFQDN(string)
```

Option 82 - relay agent information

The value for option 82 is comprised of a group of suboptions. LDRM automatically parses option 82 into a list of values where each list member corresponds to a suboption. Therefore, `getOption("relay-agent-info")` returns the value, if populated, as a list. Similarly, `setOption("relay-agent-info",value)` expects value to be a list.

When working with the suboptions, it is important to know that each one uses a different datatype. It is the user's responsibility to know the appropriate datatype for a given suboption. When writing rules that work with option 82 data, you must check that option 82 data is present (that is, check for `None`), that a given suboption is present (that is, check the list size), and that the given suboption has a value (that is, another check for `None`). This point is illustrated in the following example.

```
# Get Option 82
opt82 = dhcpIn.getOption(“relay-agent-info”)
if opt82 != None :
    RuleLib.log(“Option 82 is present”)
    if len(opt82)>0 and opt82[0] != None :
        RuleLib.log(“Opt82 Subopt1=”,opt82[0])
    if len(opt82)>1 and opt82[1] != None :
        RuleLib.log("Opt82 Subopt2=",opt82[1])
    if len(opt82)>3 and opt82[3] != None :
        RuleLib.log(“Opt82 Subopt4=, opt82[4]”)
```

Option 82 uses mixed data types. It is the user’s responsibility to extract the appropriate data type.

Important! List members are numbered starting at 0, therefore the first suboption can be found at position [0].

□

Example rules file

The following is an example of a *rules.py* file. This file contains the following rules:

- If the giaddr is 192.168.100.1, set the user-class to visitor and set the lease time to 3600 seconds (1 hour). This rule was set up to identify visitors logging in from the lobby of a building, and limits their lease time.
- If the ClientId is not blank, and the myVendorClass is Acme Handheld, set the device's MAC address to start with 00 and end with the first 10 bytes from the client ID option.

Rules file header. This text must be the same in all rules files.

```
# Use header comments to add some version identification.
# File: rules.py, version XYZ.
# Authored by Your Name, etc.


from com.lucent.qtek.ldrm.ruleApi import *
from com.lucent.qtek.ldrm.ruleApi.RuleLib import *

class rules(RuleIntf):
  def process(self,dhcpIn,dhcpOut) :
```

Rules file body.

This section is user-defined.

```
# Use giaddr to determine IP network of client.
      if dhcpIn.getGiAddr() == "192.168.100.1" :
          # Connecting from the lobby.
          dhcpOut.setOption("user-class", "student")
          dhcpOut.setOption("dhcp-lease-time", 3600)

# Uses packet receipt callout
# Mac Addr Based on ClientID and vendor-class
# Use with restoreoriginalchaddr policy in dhcpd.pcy file
      if dhcpIn.getDhcpEvent()== packetreceipt :
          opt61 = dhcpIn.getOption("dhcp-client-id")
          opt77 = dhcpIn.getOption("vendor-class")
          if opt61 is not None and opt61[0:1]!="\x00" \
             and opt77 != "Acme Handheld" :
              return
              # Normal Processing of packet
          else :
              dhcpOut.setChAddr("00:" +  opt61[1:3] + ":" \
                + opt61[3:5] + ":" + opt61[5:7] + ":" \
                + opt61[7:9] + ":" + opt61[9:11])
```

Call additional rules files. If you are using multiple rules files, uncomment the import command(s) and specify the additional rules file names to call them.

```
#use the following lines to call additional rules files
#import rule1.py
#import rule2.py
```

□

Test LDRM rules

Lucent provides a rules testing utility with LDRM. This utility takes your compiled *rules.jar* file and runs it against at least one sample DHCP packet that you create. It then displays the output on your screen.

To test your rules file, you must create at least one packet file, such as *packet.py.* Create a packet file using a standard text editor containing `dhcpIn.set<XYZ>` commands. The following is an example of standard input packet commands:

```
dhcpIn.setChAddr( “01:02:03:04:05:06” )
dhcpIn.setOp(201)
dhcpIn.setOption( “user-class”, “visitor” )
```

Refer to “Packet variables”, on page 57 for a list of values you can set when you create your packet file. The packet file specifies values for the various components of the DHCP packet against which you are testing your rules file.

> **Important!** This packet file you create is only used with the rules tester. In a production environment, these values come from the client that is requesting a DHCP lease. Refer to “LDRM rule library”, on page 64 for a sample packet file.

This command-line based utility allows you to test your rules against possible DHCP packets. By using this utility to test rules and correct any coding errors you may have made, you can prevent unpredictable results on your production system, and reduce the number of errors that appear in the DHCP log file.

Typically, rule errors in a production environment result in the DHCP packet being rejected, which causes the client to not receive a response from the DHCP server. If the client does not receive a response, the client cannot connect to the network. If you experience rule errors in a production environment, check the log file to track down the reason.

□

Test a rules.jar file

This section describes the procedure to test a *rules.jar* file.

Procedure

To test a *rules.jar* file you created in “Create rules”, on page 31, perform the following steps:

1 Create at least one packet file that reflects a typical DHCP packet. This file must have a suffix of .py. Lucent recommends creating multiple packet files to test your rules against different types of input packets. Refer to “LDRM rule library”, on page 64 for a sample packet file.

Important! You can give these files any name you want, but it must have a suffix of *.py*.

2 From a command line, type *ldrmtester.sh* This command starts the rules tester and prompts you for the name of the *rules.jar* file. This is the file you created in “Create rules”, on page 31.

3 Type *rules.jar* and press **Enter**. You are prompted for the name of an input file containing sample packet data.

4 Type the filename and press **Enter**. This is a file you created in Step 1.

The rules tester runs the compiled *rules.jar* file against the sample packet data file you provide. It displays modified packet data if your rules file runs successfully, or an error message stating the line number in the *rules.py* file where an error occurred.

5 When you are prompted for a packet file, enter the name of another packet against which you want to run your rules. The rules tester displays the input packet and the output packet.

The following is shortened sample rules tester output for the examples shown in this section. See “Rules tester output”, on page 62 for a complete output example.

```
INPUT:
======
chAddr: 01:02:03:04:05:06
Options: {77 = visitor}

OUTPUT:
======
chAddr: null
Options: {51 = 3600
          77 = student}
```

6 When you are finished running packets against your *rules.jar* file, press **Enter** when prompted for a packet name, and the rules tester ends.

END OF STEPS

□

Packet variables

For test packets, you can use the following variables to specify which DHCP callout sends information to LDRM.

For each `dhcpOut.set` command, note that strings must be in quotes, and numeric values must not be in quotes. Syntax is shown in Table 12.

For Discover, Request, and Bootp Request API callouts, the LDRM rules support the assignment of values to most of the DHCP variables in the following table. (There are some exceptions in which some variables are not used in Bootp Request API callouts, and these are identified in the rows below.) Additionally, the specified validations are performed on variable values set in the LDRM rules. (Each DHCP Option in Table 12 is described in the RFC specified in the first table at ***www.iana.org/assignments/bootp-dhcp-parameters***.) If validation fails, then `setAuthorized` is set to `off` and a Return Value of 0 is sent to the DHCP Server.

If validation fails for a Packet Receipt API callout, then `setAuthorized` is set to `off` and a Return Value of 0 is sent to the DHCP Server. Also, the specified validations are performed on variable values set in the LDRM rules.

Important! If validation fails, the following occurs:

- LDRM reports errors to LDRM.*log*
- LDRM automatically turns off setauthorization
- The DHCP server discards the packet and writes an entry to *dhcpserver.log*

Table 12 Packet Variables

Variable	Description	Validation	Usage
ChAddr	Field in DHCP Message Client hardware address	String. MAC address. Six 2-character strings, each separated by a colon. For example: `01:02:03:04:05:06`	Invoked with `dhcpIn.setChAddr("<`*`MAC Address`*`>")`
CiAddr	Field in DHCP Message Client IP address; only filled in if client is in BOUND, RENEW or REBINDING state and can respond to ARP requests	String. IP address in standard dotted decimal format.	Invoked with `dhcpIn.setCiAddr("<`*`IP Address`*`>")`

Variable	Description	Validation	Usage

DhcpEvent	Specifies the callout the DHCP server used to send a message to LDRM. See “Configure the DHCP Service for LDRM”, on page 20 for additional information.	Values are:
		`packetreceipt`
		`discover`
		`request`
		`bootprequest`
			Invoked with `dhcpIn.setDhcpEvent(<callout name>)`

Flags	Field in DHCP Message To work around some clients that cannot accept IP unicast datagrams, DHCP uses the 'flags' field.	Number. The leftmost bit is defined as the BROADCAST (B) flag. The remaining bits of the flags field are reserved for future use. They must be set to zero by clients and are ignored by servers and relay agents.	Invoked with `dhcpIn.setFlags(1)`
GiAddr	Field in DHCP Message Relay agent IP address, used in booting via a relay agent	String. IP address in standard dotted decimal format.	Invoked with `dhcpIn.setGiAddr("<IP Address>")`
HLen	Field in DHCP Message Hardware address length	Value must be 12.	Invoked with `dhcpIn.setHLen(12)`
Hops	Field in DHCP Message	Integer. Client sets to zero, optionally used by relay agents when booting via a relay agent	Invoked with `dhcpIn.setHops(<integer>)`
HType	Field in DHCP Message Hardware address type, 1 Ethernet (10Mb)	Value must be 1.	Invoked with `dhcpIn.setHType(1)`
Secs	Field in DHCP Message	Integer. Filled in by client, seconds elapsed since client began address acquisition or renewal process.	Invoked with `dhcpIn.setSecs(<number of seconds>)`
XID	Field in DHCP Message Transaction ID.	A random number chosen by the client, used by the client and server to associate messages and responses between a client and a server.	Invoked with `dhcpIn.setXID(<transaction ID>)`

Variable	Description	Validation	Usage
requested-IP-addr	DHCP Option 50	Formatted in standard dotted-decimal notation. Prohibited in Bootp Request callouts.	Invoked using `dhcpIn.setOption("requested-IP-addr", "<IP Address>")` Set by client/LDRM, and used by server to select IP address.
DHCP - lease-time	DHCP Option 51	Integer. Prohibited in Bootp Request callouts. A value of `-1` indicates an indefinite lease. Set by client/LDRM, and used by server as requested leasetime. This option is sent to client in offer/ack if HonorRequestedLease Time policy is enabled, or the requested value is less than what would be assigned from configuration data.	Invoked using `dhcpIn.setOption("DHCP-lease-time", <seconds in lease>)`
DHCP-message-type	DHCP Option 53	Integer.	Invoked with `dhcpIn.setOption("DHCP-message-type", <#>)`
DHCP-server-id	DHCP Option 54	IP Address.	Invoked with `dhcpIn.setOption("DHCP-server-id", "<IP Address>")`
parameter-request-list	DHCP Option 55	Binary string.	Invoked with `dhcpIn.setOption("parameter-request_list", "<string>")`
vendor-class	DHCP Option 60	String.	Invoked with `dhcpIn.setOption("vendor-class", "<vendor class>")` Not used for Bootp
dhcp-client-id	DHCP Option 61	Binary string.	Invoked with `dhcpIn.setOption("dhcp-client-id", "<string>")`
user-class	DHCP Option 77	String. Not used for Bootp.	Invoked with `dhcpIn.setOption("user-class", "<user class>")`

Variable	**Description**	**Validation**	**Usage**
client-FQDN	DHCP Option 81	String whose length is between 3 and 255. Not used for Bootp. See LDRM rule library on page 64 for more information.	Invoked with `dhcpIn.setOption(client-FQDN”, “<`*`string`*`>”)`
relay-agent-info	DHCP Option 82	List of binary strings. Contains one or more sub-options.	Invoked with `dhcpIn.setOption(“relay-agent-info”, [“`*`string1`*`”, “`*`string2`*`”, . . .])`
subnet-selection	DHCP Option 118	IP Address.	Invoked with `dhcpIn.setOption(“subnet-selection”, “<`*`IP Address`*`>”)`

□

Example DHCP packet file

The following is an example of a DHCP packet file that you can use for testing. This packet sets all possible fields. Testing of rules may not require using all fields. You can create input packets with any subset of fields.

```
dhcpIn.setChAddr( “01:02:03:04:05:06” )
dhcpIn.setOp(1)
dhcpIn.setHType(1)
dhcpIn.setHLen(6)
dhcpIn.setHops(204)
dhcpIn.setXID(205)
dhcpIn.setSecs(1000)
dhcpIn.setFlags(\x8000)
dhcpIn.setCiAddr(“1.2.3.4”)
dhcpIn.setGiAddr(“2.3.4.5”)
dhcpIn.setOption( “user-class”, “student” )
dhcpIn.setOption( “routers”, [“10.20.30.40”,”50.60.70.80”] )
dhcpIn.setOption( “DHCP-message-type”, 1 )
```

Any values not specified in this file default to 0.

□

Rules tester output

Table 13 is an example of how running the rules tester appears on your screen:

Table 13 Rules tester output example

Explanation	**Example**

Start the LDRM rules tester.	`ldrmtester.sh rules.jar`
	
	`dhcpOpts.ParseConfig:   Successfully loaded 71 option`
	`  specs.`
	
	`script.Loader:  Loading rules...`
	`script.Loader:  Rules loaded.`

Enter packet name.	`Enter file containing a DHCP packet: packet.py`

This is a trace of the LDRM API calls made by the packet file (*packet.py*).	`ruleApi.DhcpWriter:     setChAddr(01:02:03:04:05:06)`
	`ruleApi.DhcpWriter:     setOp(201)`
	`ruleApi.DhcpWriter:     setHType(1)`
	`ruleApi.DhcpWriter:     setHLen(6)`
	`ruleApi.DhcpWriter:     setHops(204)`
	`ruleApi.DhcpWriter:     setXID(205)`
	`ruleApi.DhcpWriter:     setSecs(1000)`
	`ruleApi.DhcpWriter:     setFlags(2000)`
	`ruleApi.DhcpWriter:     setCiAddr(1.2.3.1)`
	`ruleApi.DhcpWriter:     setGiAddr(192.168.100.1)`
	`ruleApi.DhcpWriter:     setOption('user-class',visitor)`
	`ruleApi.DhcpWriter:`
	`  setOption('routers',[10.20.30.40,50.60.70.80])`

Explanation	Example

Rules tester displays the content of the packet file you specify.	`INPUT:`
	`======`
	`dhcpEvent: DISCOVER`
	`bootFile: null`
	`tftpServer: null`
	`op: 201`
	`hType: 1`
	`hLen: 6`
	`hops: 204`
	`xid: 205`
	`secs: 1000`
	`flags: 2000`
	`ciAddr: 1.2.3.1`
	`giAddr: 192.168.100.1`
	`chAddr: 01:02:03:04:05:06`
	`authorized: true`
	`Options: {77 = visitor`
	`          3 = [10.20.30.40, 50.60.70.80]}`

Press **Enter** when you are ready to proceed.	`Push return to process:`
	`<trace of LDRM calls>`

Rules tester displays content of dhcpOut after the rules were executed.	`OUTPUT:`
	`======`
	`dhcpEvent: DISCOVER`
	`bootFile: null`
	`tftpServer: null`
	`op: 0`
	`hType: 0`
	`hLen: 0`
	`hops: 0`
	`xid: 0`
	`secs: 0`
	`flags: 0`
	`ciAddr: null`
	`giAddr: null`
	`chAddr: null`
	`authorized: true`
	`Options: {51 = 3600`
	`          77 = student}`

Specify another packet, or press **Enter** to exit.	`Enter file containing a DHCP packet:`
	`Exit.`

LDRM rule library

The input and output variables that LDRM uses are part of library module called RuleLib. The following tables are a quick reference as to which you can use in rules, and which in input packets. Use the following commands to invoke the variables in your *rules.py* file or your input packet file.

Table 14 Variables in rules and input packets

Variable name	Set in rules	Get in rules	Set in input test packets
Authorized	✓		
BootFile	✓		✓
ChAddr		✓	✓
CiAddr		✓	✓
DhcpEvent		✓	✓
Flags		✓	✓
HLen		✓	✓
Hops		✓	✓
HType		✓	✓
Op		✓	✓
Secs		✓	✓
SuggestedSubnet	✓		
TftpServer	✓		✓
XID		✓	✓

Table 15 DHCP options in rules and input packets

DHCP option	Option number	Set in rules	Get in rules	Set in input test packets
time-offset	DHCP Option 2	✓		
routers	DHCP Option 3	✓		
time-servers	DHCP Option 4	✓		
name-servers	DHCP Option 5	✓		
domain-name-servers	DHCP Option 6	✓		
log-servers	DHCP Option 7	✓		
cookie-servers	DHCP Option 8	✓		
lpr-servers	DHCP Option 9	✓		
impress-servers	DHCP Option 10	✓		
resource-location-servers	DHCP Option 11	✓		
host-name	DHCP Option 12	✓		

DHCP option	Option number	Set in rules	Get in rules	Set in input test packets
boot-size	DHCP Option 13	✓		
merit-dump	DHCP Option 14	✓		
domain-name	DHCP Option 15	✓		
swap-server	DHCP Option 16	✓		
root-path	DHCP Option 17	✓		
extensions-path	DHCP Option 18	✓		
IP-forwarding	DHCP Option 19	✓		
non-local-source-routing	DHCP Option 20	✓		
policy-filter	DHCP Option 21	✓		
max-dgram-reassembly	DHCP Option 22	✓		
default-IP-TTL	DHCP Option 23	✓		
path-MTU-aging-timeout	DHCP Option 24	✓		
all-subnets-local	DHCP Option 27	✓		
broadcast-address	DHCP Option 28	✓		
perform-mask-discovery	DHCP Option 29	✓		
mask-supplier	DHCP Option 30	✓		
router- discovery	DHCP Option 31	✓		
router-solicitation-address	DHCP Option 32	✓		
static-routes	DHCP Option 33	✓		
trailer- encapsulation	DHCP Option 34	✓		
arp-cache timeout	DHCP Option 35	✓		
ieee802-3 encapsulation	DHCP Option 36	✓		
default-tcp-ttl	DHCP Option 37	✓		
tcp-keepalive- interval	DHCP Option 38	✓		
tcp-keepalive-garbage	DHCP Option 39	✓		
nis-domain name	DHCP Option 40	✓		
nis-servers	DHCP Option 41	✓		
ntp-servers	DHCP Option 42	✓		
vendor-specific	DHCP Option 43	✓	✓	
netbios-name-servers	DHCP Option 44	✓		
netbios-dd-servers	DHCP Option 45	✓		
netbios-node-type	DHCP Option 46	✓		
font-servers	DHCP Option 48	✓		
x-display-manager	DHCP Option 49	✓		
requested-IP-addr	DHCP Option 50		✓	✓
dhcp-lease-time	DHCP Option 51		✓	✓
dhcp-message-type	DHCP Option 53		✓	✓
dhcp-server-id	DHCP Option 54		✓	✓

DHCP option	**Option number**	**Set in rules**	**Get in rules**	**Set in input test packets**
parameter-request-list	DHCP Option 55		✓	✓
dhcp-renewal-time	DHCP Option 58	✓		
dhcp-rebinding-time	DHCP Option 59	✓		
vendor-class	DHCP Option 60	✓	✓	✓
dhcp-client-id	DHCP Option 61		✓	✓
nis+ domain	DHCP Option 64	✓		
nis+ servers	DHCP Option 65	✓		
mobile-ip-home-agents	DHCP Option 68	✓		
smtp-servers	DHCP Option 69	✓		
pop3-servers	DHCP Option 70	✓		
nntp-servers	DHCP Option 71	✓		
www-servers	DHCP Option 72	✓		
finger-servers	DHCP Option 73	✓		
irc-servers	DHCP Option 74	✓		
streettalk-servers	DHCP Option 75	✓		
dhcp-stda-servers	DHCP Option 76	✓		
user-class	DHCP Option 77	✓	✓	✓
client-FQDN	DHCP Option 81	✓	✓	✓
relay-agent-info	DHCP Option 82		✓	✓
novell-nds-servers	DHCP Option 85	✓		
ipv4-auto-configuration	DHCP Option 116	✓		
subnet-selection	DHCP Option 118		✓	✓

The following output commands (used in rules) have additional considerations:

- `dhcpOut.setAuthorized( boolean value )`

  If a value is assigned to a DHCP variable in an LDRM rule, and it fails its validation, then the `Authorize` parameter is set to False.

  When one of the above field values is unspecified, a 0 is returned for numeric values, or two double quotes (“ ”) are returned for string values.

- `dhcpOut.setOption( String name,value )`

  If an option is unspecified, `None` is returned. `None` indicates no value for the option, and must not be used.

  Important! If an option is specified in the input DHCP client packet, LDRM cannot override it to be unspecified. LDRM can only override an option value to a non-null value.

A DHCP packet contains both fielded values and options. Fielded values are always present, and options are just that, optional. Therefore, options behave differently from fielded values when no value has been assigned. Fielded values always have some value, even if it is zero, or just an empty string (for example, “ ”). In contrast, when an option is missing it has no value and is represented as `None`.

Using a variable containing `None` in a simple comparison is acceptable, but attempting to reference one of its methods causes a Jython execution error. In the following line of code, the option `example` may contain a string value, but in this case is not populated:

```
var = dhcpIn.getOption(“example”)      # var will be set to “None”.
```

Compares with the `None` value are allowed; the following is false:

```
    if var == “student” :
```

Checking for set membership is also permitted, as shown in the following:

```
    if var in [“item1”,“item2”] :
```

But trying to use one of var’s string methods when var has `None` is illegal and results in a runtime error message containing a verbose stacktrace:

```
    if var.startswith(“stu”) :
```

Therefore, a good programming practice is to first check options for `None`:

```
    if var != None  and  var.startswith(“stu”) :
```

See “Rule variables”, on page 35 and “Packet variables”, on page 57 for specific instructions on which of the above commands you must use to get or set a specific variable.

Multi-threading in LDRM

Each DHCP service thread opens its own socket connection to LDRM. As LDRM accepts each new connection, it creates a new thread and a rule processor instance. Therefore, the number of simultaneous client connections LDRM can process is controlled by the number of threads configured in the DHCP service.

When designing LDRM rules, you must ensure that the rules do not attempt to share data between threads or retain data between transactions. Your rule variables can only be used within the rule class itself; attempts to use language features such as "global" will fail. This ensures "Thread Safety". As each new thread and rule processor instance is created in LDRM it starts executing your rules, so there is no current way for global variables to be initialized only once.

In summary, LDRM is designed to be stateless. Do not attempt multithreading in LDRM rules despite Jython's capability to do so. Do not use global variables.

□

5 Troubleshooting

Overview

Purpose

This chapter contains common error messages you receive when testing rules, and suggested resolutions. It also contains information you can use to determine how well LDRM is running.

Contents

This information presents the following topics.

☐

Sample files

The error messages in this section are generated against the following rules file and packet file.

Rules File

The following rules file has correct syntax. All the error messages that follow are generated from modifications to the following correct file:

```
# File: rules.py, version XYZ.
# Authored by Your Name, etc.

from com.lucent.qtek.ldrm.ruleApi import *
from com.lucent.qtek.ldrm.ruleApi.RuleLib import *

class rules(RuleIntf):
  def process(self,dhcpIn,dhcpOut) :
      # Use giaddr to determine IP network of client.
      if dhcpIn.getDhcpEvent() != packetreceipt :
         if dhcpIn.getGiAddr() == "192.168.100.1" :
             # Connecting from the lobby.
             dhcpOut.setOption("user-class", "visitor")
             dhcpOut.setOption("DHCP-lease-time", 3600)

      # Use LDRM to set MAC address.
      if dhcpIn.getDhcpEvent() == packetreceipt :
          myDHCPclientId = dhcpIn.getOption("dhcp-client-id")
          myVendorClass = dhcpIn.getOption("vendor-class")
          if myDHCPclientId is not None  and  \
                 myVendorClass == "Acme Handheld" :
              # Set Client Hardware Address (i.e. MAC)
              # Extract bytes from string using [start:end]
              dhcpOut.setChAddr( "08:07:06" +
                                 ":" + myDHCPclientId[0:2] +
                                 ":" + myDHCPclientId[3:5] +
                                 ":" + myDHCPclientId[6:8] )
```

Packet File

The following packet file has correct syntax. All the error messages that follow are generated from modifications to the following correct file:

```
dhcpIn.setChAddr( "01:02:03:04:05:06" )
dhcpIn.setOp(201)
dhcpIn.setHType(1)
dhcpIn.setHLen(6)
dhcpIn.setHops(204)
dhcpIn.setXID(205)
dhcpIn.setSecs(1000)
dhcpIn.setFlags(2000)
dhcpIn.setCiAddr("1.2.3.4")
dhcpIn.setGiAddr("192.168.100.1")
dhcpIn.setOption( "user-class", "student" )
dhcpIn.setOption( "routers", ["10.20.30.40","50.60.70.80"])
```

□

LDRMCOMPILER errors

The following are some standard error messages that can occur when your *rules.py* file is compiled into a *rules.jar* file. Each message is followed by a possible resolution:

Missing Punctuation

```
SyntaxError: Lexical error at line <#>, column <#>.  Encountered: "\n" (10), after : ""
```

Check the line and column number to look for bad syntax, for example a missing quotation mark.

```
File "<string>", line <#> if dhcpIn.getGiAddr( == "192.168.100.1" :
                                    ^SyntaxError: invalid syntax
```

In the above example, the `dhcpIn.getGiAddr` command is missing a closing parentheses

Incorrect Line Break

```
File "<string>", line <#>
                  if myDHCPclientId is not None  and
```

In the above example, the line ends with the word `and`. The resolution is that the following line is a continuation of the displayed line. However, the continuation character at the end of the line, a backslash (\) is missing.

```
SyntaxError: Lexical error at line <#>, column <#>.  Encountered: " " (32), after :
  "\\"
```

In the above example, the problem is that there is a blank space after the backslash used as a line continuation character (\). If you use the backslash at the end of a line, you cannot have any characters after it, including blank spaces.

Incorrect Operator

```
File "<string>", line <#>
             if dhcpIn.getGiAddr() = "192.168.100.1" :
```

The problem with the above example is with the “if” statement. You must have a double equal sign (==) between the `dhcpIn.getAddr()` command and the value. The error was caused by a single equal sign being used instead of a double equal sign.

□

LDRMTESTER errors

The following errors occur when you run LDRMTESTER against your compiled *rules.jar* file.

> **Important!** You can receive errors from the LDRMTESTER even if you did not receive errors from the LDRMCOMPILER. Be sure to fix all errors from the LDRMCOMPILER before running the LDRMTESTER.

Incorrect Value Type

The first set of error messages, shown below, occurred because LDRM expected a numeric value, but found a string in its place. This occurred because there are quotation marks around the value for lease time in the *rules.py* file. When quotation marks are put around any value, even if the value is a number, LDRM treats the value as a string.

To determine where in the *rules.py* file the error occurred, check the error messages for the lines that contain the word `Exception`. Note that, in the following example, the numeric value `3600` does not appear in quotes. However, in the *rules.py* file that generated this message, the value `3600` was in quotes. Therefore, you can use the error messages to locate the problem, but they do not necessarily show you the exact syntax that caused it. You must view the *rules.py* file to see that.

```
ruleApi.DhcpReader:     getGiAddr() = 192.168.100.1
ruleApi.DhcpWriter:     setOption('user-class',visitor)
ruleApi.RuleApiException:       DhcpWriter.setOption('DHCP-lease-time',3600): Numeric
  value required: java.lang.String(3600)
.
.
.
com.lucent.qtek.ldrm.ruleApi.RuleApiException: DhcpWriter.setOption('DHCP-lease-
  time',3600): Numeric value required: java.lang.String(3600)
.
.
.
Caused by: java.lang.NumberFormatException: Numeric value required:
  java.lang.String(3600)
.
.
.
ApiException: DhcpWriter.setOption('DHCP-lease-time',3600): Numeric value required:
  java.lang.String(3600)
```

The following output shows what happens if the rule had used the correct syntax. Note that option 51 is set to `3600`.

```
OUTPUT:
dhcpEvent: DISCOVER
bootFile: null
tftpServer: null
op: 0
hType: 0
hLen: 0
hops: 0
xid: 0
secs: 0
flags: 0
ciAddr: null
giAddr: null
chAddr: null
authorized: true
Options: {51 = 3600
          77 = visitor}
```

Incorrect Command

The following output occurs when the wrong command is used in a rule. The line that starts `AttributeError` names the command that must be corrected.

```
Push return to process:
Traceback (innermost last):
  File "/fs1/home/userdir/rules.py", line 0, in process
AttributeError: setGiAddr
```

The cause of this output is shown below. The `dhcpIn` command must have a “`get`” instead of a “`set`” after the period. The incorrect example in the rule below caused the error message.

```
class rules(RuleIntf):
  def process(self,dhcpIn,dhcpOut) :
      # Use giaddr to determine IP network of client.
      if dhcpIn.setGiAddr() == "192.168.100.1" :
```

Invalid Option

The following error messages occur when an option is invalid, in this case due to a misspelling or typing error.

```
ruleApi.DhcpReader:     getGiAddr() = 192.168.100.1
ruleApi.RuleApiException:       DhcpWriter.setOption('user-clas',visitor):
  Illegal option name: user-clas
Traceback (innermost last):
  (no code object) at line 0
com.lucent.qtek.ldrm.ruleApi.RuleApiException: DhcpWriter.setOption('user-
  clas', visitor): Illegal option name: user-clas
.
.
.
Caused by: java.lang.IllegalArgumentException: Illegal option name: user-clas
.
.
.
com.lucent.qtek.ldrm.ruleApi.RuleApiException:
  com.lucent.qtek.ldrm.ruleApi.Rule
ApiException: DhcpWriter.setOption('user-clas',visitor): Illegal option name:
  user-clas
```

To determine where in the *rules.py* file the error occurred, check the error messages for the lines that contain the word `Exception`. Note that, in this example, the option name `user-clas` is missing an s and is therefore not recognized as a legal option.

☐

LDRM Status

When to use

The LDRM Status screen displays a snapshot of LDRM's runtime status. Check this screen if LDRM is not performing as expected. This screen can help you perform basic troubleshooting by letting you check system status, memory usage, the name, date, and timestamp of your *rules.jar* file, and assorted error messages.

Description

You can access this page by entering the following URL into any web browser's address field.

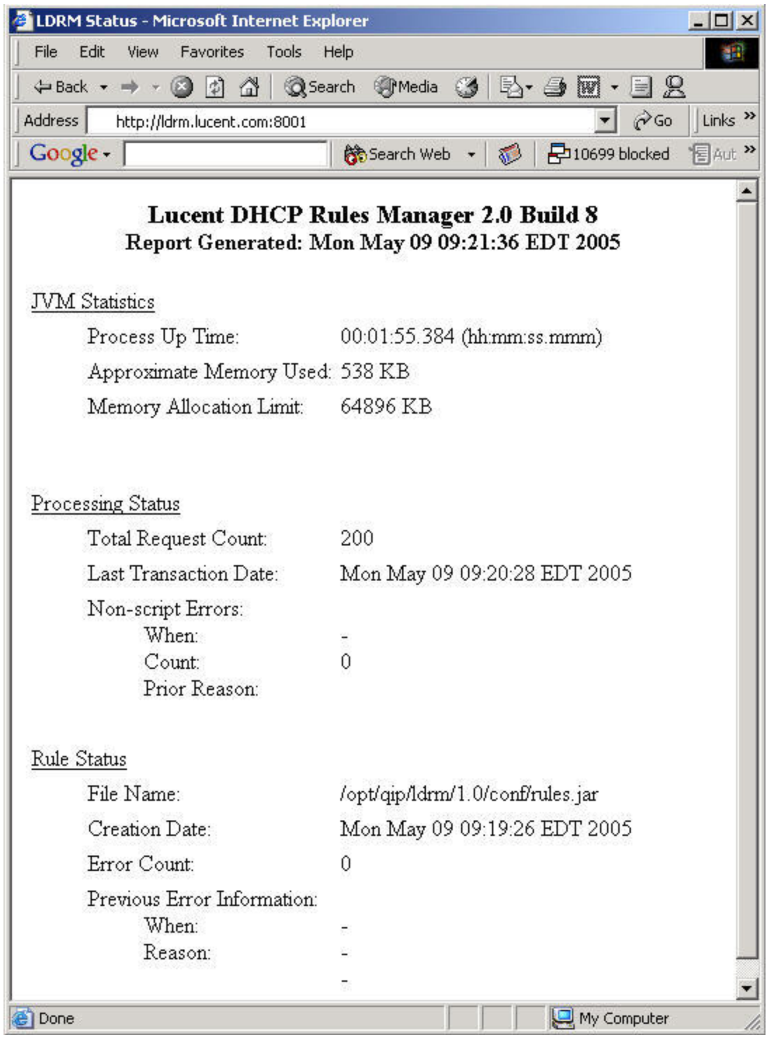
http://<system_name>:<port_number>

`System_name` is the name of the LDRM server whose performance you want to check.

`Port_number` is the port number specified in the *ldrm.properties* file as the `ldrm.admin.port.` See "Configure the LDRM Service", on page 24 for more information.

Figure 6 shows a sample LDRM Status screen.

Figure 6 LDRM Status screen


LDRM Status - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Media
Address http://ldrm.lucent.com:8001
Go Links
Google Search Web 10699 blocked
Done
My Computer

Lucent DHCP Rules Manager 2.0 Build 8
Report Generated: Mon May 09 09:21:36 EDT 2005

JVM Statistics

Process Up Time:	00:01:55.384 (hh:mm:ss.mmm)
Approximate Memory Used:	538 KB
Memory Allocation Limit:	64896 KB

Processing Status

Total Request Count:	200
Last Transaction Date:	Mon May 09 09:20:28 EDT 2005
Non-script Errors:	
When:	-
Count:	0
Prior Reason:	

Rule Status

File Name:	/opt/qip/ldrm/1.0/conf/rules.jar
Creation Date:	Mon May 09 09:19:26 EDT 2005
Error Count:	0
Previous Error Information:	
When:	-
Reason:	-
	-

Table 16 describes the values on the LDRM Status screen:

Table 16 LDRM Status screen values

Field	**Description**
JVM Statistics	
Process Up Time	Amount of time LDRM has been up and running.
Approximate Memory Used	Amount of memory LDRM is currently using.

Field	**Description**
Memory Allocation Limit	Total amount of memory LDRM has available to it.
Processing Status	
Total Request Count	Number of packets that have moved between LDRM and DHCP. Increases with the number of LDRM callouts you have enabled.
Last Transaction Date	Most recent date and time that a packet has moved between LDRM and DHCP.
Non-script errors	
When	Date and time of the most recent non-script error.
Count	Number of errors LDRM has generated. Normally should be 0. If an error is reported here, it may indicate that LDRM may not be able to continue with any processing.
Prior Reason	Error message describing the reason for the error.
Rule Status	
File Name	Full path and name of the .jar file containing the compiled rules.
Creation Date	Date and time the *rules.jar* file was copied into LDRM's *conf* directory.
Error Count	Number of errors caused by problems during rule processing. Depending on the nature of the rule error, other requests may still be processed successfully. Nonetheless, a robust set of rules should not experience faults.
Previous Error Information	
When	Date and time of the most recent LDRM error.
Reason	Error message describing the reason for the error. Error messages listed here are generated by LDRM, but are the same as those generated by the LDRM rules tester. For information on these error messages, see “LDRMTESTER errors”, on page 73.

□

DHCP LDRM Callout Statistics

When to use

The DHCP LDRM Callout Statistics file contains performance data for each of the DHCP server callouts to the LDRM service. For each callout type, the server tracks the minimum and maximum response times, as well as the total cumulative response time and the number of callouts made. This data can be used to determine the additional time required to process each type of DHCP message through the LDRM service. This file is generated periodically based on the value of the StatisticsDumpFrequency policy, contained in the *LDRMCallout.pcy.file*. See "LDRMCallout.pcy file", on page 22 for more information.

Example

```
LDRM Callout Stats Init Time = Fri Apr 08 10:27:12.505
LDRM Packet Receipt Callouts = 1500
       Packet Receipt Minimum Response Interval = 0 milliseconds
       Packet Receipt Maximum Response Interval = 40 milliseconds
       Packet Receipt Total Response Interval = 3778 milliseconds

LDRM Discover Callouts = 250
       Discover Minimum Response Interval = 10 milliseconds
       Discover Maximum Response Interval = 20 milliseconds
       Discover Total Response Interval = 2102 milliseconds

LDRM Request Callouts = 250
       Request Minimum Response Interval = 10 milliseconds
       Request Maximum Response Interval = 21 milliseconds
       Request Total Response Interval = 1874 milliseconds

LDRM BootpRequest Callouts = 0
       Bootp Minimum Response Interval = 0 milliseconds
       Bootp Maximum Response Interval = 0 milliseconds
       Bootp Total Response Interval = 0 milliseconds

LDRM Ack Callouts = 0
```

☐

LDRM Tracing

When to use

If you cannot isolate problems using the *ldrmtester*, there are three levels of runtime tracing available: from DHCPD, the LDRM callout, and LDRM itself. To best observe how LDRM is affecting the DHCP packet, start with LDRMCallout tracing, then proceed to LDRM Engine Tracing and finally try DHCPD tracing. These tracing methods are described below.

LDRMCallout tracing

LDRM callout tracing is enabled by configuring the *Debug* policy in *$QDHCPCONFIG/LDRMCallout.pcy.* Output is written to *$QIPHOME/log/LDRMCallout.log*. For information about the callout.pcy file, see "LDRMCallout.pcy file", on page 22. The allowable values for the Debug parameter areas follows:

Value	Definition
0	Minimal logging
1	Event logging (default)
2	Full logging

Note that the *MaxDebugFileSize*, which is the size at which *LDRMCallout.log* is reinitialized, defaults to `1,000,000`, but is a modifiable value.

LDRM engine tracing

Another debugging option is to enable tracing in the LDRM Rule engine itself. You can configure this in *$LDRMHOME/conf/ldrm-log4j.properties* by setting `log4j.category.com.lucent.qtek.ldrm=DEBUG` (or `INFO`). `"INFO"` provides similar tracing as the LDRM rule tester and `"DEBUG"` provides additional packet tracing. Output is written to *$QIPHOME/log/ldrm.log*. Logging is based on Java's "log4j" package; therefore the new trace level is automatically detected without restarting the software.

DHCPD tracing

This is the standard DHCP tracing, configured in *$QDHCPCONFIG/dhcpd.pcy*, by setting `"debug=all"`. Output is written to *$QIPHOME/log/dhcpd.log*. This trace records the hex dump of the DHCP packet.

□

Index