Cisco Prime Network 4.3.1 Customization Guide

Cisco Systems, Inc.
www.cisco.com

Cisco has more than 200 offices worldwide.
Addresses, phone numbers, and fax numbers
are listed on the Cisco website at
www.cisco.com/go/offices.

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

CONTENTS

PART 1 **Overview of the Customization Tools**

CHAPTER 1 **Prime Network Customization Overview** 1-1

Tools for Extending Prime Network 1-2

PART 2 **Extending the Information Model**

CHAPTER 2 **Modeling and Displaying Additional NE Properties** 2-1

What are Soft Properties? 2-1

Steps for Adding New NE Property Information to the Inventory 2-3

Displaying a New Property in the Inventory Window 2-4

Adding a New Inventory Table Containing SNMP get Details 2-11

Making the Soft Property Available to Other NEs 2-14

CHAPTER 3 **Adding Support for New Devices, Software Versions and Modules Using the VCB** 3-1

VCB Overview 3-1

Accessing the VCB 3-2

Comparison of Generic SNMP VNEs, U-VNEs, and Developed VNEs 3-3

Identifying Supported Devices 3-6

Adding Support for Additional Device Types by Creating U-VNEs 3-6

Methods for Creating U-VNEs—Overview 3-7

Creating a U-VNE Using the GenericUVNE Template 3-7

Create a U-VNE by Cloning 3-8

Defining a Device Type to Determine the GUI Representation of VNE Drivers 3-9

Adding Support for Additional Software Versions 3-11

Add Support for Additional Modules 3-12

Adding Support for a New Standard Module 3-12

Adding Support for a New Pluggable Module 3-13

Testing and Certifying VCB Customizations 3-14

Preparing to Test Your VCB Customizations 3-15

Testing and Certifying U-VNEs 3-15

Setting Up the U-VNE Test Environment 3-16

Performing the U-VNE Tests 3-18

Testing and Certifying Module Customizations 3-21
Testing and Certifying Event Customizations 3-22
Troubleshooting VCB Customizations 3-23
Deploying VCB Customizations to Your Production Environment 3-24
Viewing Existing VCB Customizations in the Registry 3-24
Exporting VCB Customizations to Another Gateway 3-26
Importing VCB Customizations from Another Gateway 3-26
Deleting VNE Customizations from the Registry 3-27

PART 3 Extending Device Configuration Capabilities

CHAPTER 4 Creating Commands and Command Scripts to Perform Device Configuration Operations 4-1
How is Command Manager Different From Command Builder? 4-1
Checking Global Command Settings Before You Begin Using Command Manager 4-2
Using Command Manager to Create and Execute Device Configuration Command Scripts 4-3
What is Command Manager? 4-3
Steps for Creating and Deploying Commands with Command Manager 4-4
Command Manager Window and Repository 4-5
Creating and Testing a Command or Command Sequence 4-7
Creating a New Package 4-7
Creating a New Command 4-8
Creating a Command Sequence 4-15
Previewing a Command Script and Checking Command Job Results 4-16
Running Commands Using Input Stored in a File 4-17
Controlling Command Manager Jobs 4-18
Moving, Importing, and Exporting Commands and Packages 4-18
Deleting Commands and Packages 4-19
Using Command Builder to Create Device Configuration Command Scripts 4-20
What is Command Builder? 4-20
Steps for Defining a New Command 4-21
Command Builder Window 4-21
Creating a Command Using Command Builder 4-23
Defining a Combo Field Type for a Command Builder Command 4-29
End-to-End Example of Creating a Macro Language Command 4-30
Testing and Publishing Commands 4-34
Testing the Command Locally Before Deploying It to Other NEs 4-34
Publishing Commands to Other NEs 4-34

Exporting Commands with Command Builder 4-37
Importing Commands with Command Builder 4-37
Deleting Commands 4-38

CHAPTER 5 Deploying Activation Workflows to Devices Using Transaction Manager 5-1
What is Transaction Manager? 5-1
Setup Tasks for Transaction Manager 5-2
Steps for Creating and Deploying Transactions 5-2
Transaction Manager Window and Repository 5-3
Creating Transactions Using the Prime Network XDE SDK 5-4
Importing Transactions Into Prime Network 5-4
Previewing and Executing Transactions 5-5
Running Transactions Using Input Stored in a File 5-7
Finding and Controlling Transaction Jobs 5-8

PART 4 Extending and Customizing How Events and Faults are Managed

CHAPTER 6 Adding Support for New Events Using the VCB 6-1
Enabling Support for Unsupported Traps 6-1
Adding Support for a Trap 6-2
Adding Support for Multiple Traps Simultaneously 6-7
Enabling Support for a Custom Syslog 6-10
Parsing Rule Group Inheritance Structure 6-13
Testing Your New Events 6-14
Customizing Events 6-15
Changing the Severity of an Event Subtype 6-15
Dropping an Event 6-15
Restoring a Dropped Event 6-16
Restoring a System Default Event 6-16
Deleting Events 6-16
Displaying MIB Names Instead of Numbers in Trap Details 6-17
Adding Recommended Actions to Event Details 6-17
Adding Parameter Values to Event Descriptions 6-19

CHAPTER 7 Adding New Threshold-Crossing Alarms 7-1
Creating a New Threshold-Crossing Alarm 7-1

PART 5 Extending the GUI Client

CHAPTER 8 Adding External Launch Points to the GUI Client 8-1

Overview: Adding External Launch Points 8-1

Adding an External Launch Point to the GUI Client 8-3

Identifying the IMO Context and Properties 8-3

Creating BQL Scripts 8-4

Defining Soft Property Parameters for a BQL Script 8-9

Running BQL Scripts 8-10

Enabling Automatic Download of BQL Scripts to Clients 8-10

Deleting an External Launch Point 8-10

Sample BQL Scripts for Launching External Applications 8-11

PART 6 Appendixes

APPENDIX A Soft Properties: Supported Parsing Operators, Rules, and Regular Expressions A-1

Supported Parsing Rules and Operators A-1

Supported Regular Expressions A-3

APPENDIX B VCB: CLI Commands and Template Reference B-1

VCB CLI Reference: vcb and vcb sitechanges Commands B-1

vcb B-1

vcb sitechanges B-3

VCB CLI Reference: U-VNEs Commands B-3

vcb uvne add B-3

vcb uvne view B-6

vcb uvne modify B-9

vcb uvne delete B-11

VCB CLI Command Reference: Device Types Commands B-12

vcb devicetype add B-12

vcb devicetype view B-14

vcb devicetype modify B-15

vcb devicetype delete B-16

VCB CLI Reference: Standard and Pluggable Modules Commands B-17

VCB CLI Command Reference: Standard Modules B-18

vcb module add B-18

vcb module view B-20

vcb module modify B-22

vcb module delete B-24

VCB CLI Command Reference: Pluggable Modules B-25

vcb pluggablemodule add B-26

vcb pluggablemodule view B-28

vcb pluggablemodule modify B-29

vcb pluggablemodule delete B-30

VCB CLI Reference: Events Commands B-31

VCB CLI Reference: vcb event Commands B-31

vcb event add B-31

vcb event view B-34

vcb event modify B-37

vcb event delete B-39

VCB CLI Reference: vcb eventparsingrules Commands B-40

vcb eventparsingrules add B-41

vcb eventparsingrules view B-43

vcb eventparsingrules modify B-45

vcb eventparsingrules delete B-46

VCB CLI Reference: vcb eventpattern Commands B-47

vcb eventpattern add B-47

vcb eventpattern view B-49

vcb eventpattern modify B-50

vcb eventpattern delete B-52

VCB CLI Reference: vcb eventarg Command B-53

vcb eventarg view B-53

Troubleshooting Event Customization Using the VCB CLI B-53

VCB Template Reference B-55

U-VNE Templates B-56

GenericUVNE Template B-56

Module Templates B-65

Module Groups and Module Specification Files B-65

Module Templates by Technology B-69

Event Templates B-79

Terminology Used in Event Templates B-79

Supported Interface Types B-80

Event Templates Functional Summary B-80

Event Templates Input Summary—Required and Optional Input B-82

Event Identification Templates B-84

Unique ID Templates—for Traps Only B-86

Event Subtype Templates B-87

Entity ID Templates—for Traps Only B-89

Entity Key Templates B-90

Prime Network Event Templates B-92

APPENDIX C **Command Manager and Command Builder: Macro Language and Beanshell Reference** C-1

Prime Network Macro Language C-1

What Are Prime Network Macro Language Scripts? C-1

Properties Available from the IMO Context C-2

Specifying and Using Parameters C-2

User-Defined Parameters C-3

Multiple Formats for IP Subnet Parameters C-4

Built-In Parameters C-4

Supported Pragmas C-4

Success C-5

Fail C-5

Prompt C-6

Partial Prompt C-7

Rollback C-7

Activity C-7

Enum C-8

Example C-8

Command Script C-8

Rollback Script C-9

Running the Script C-10

BeanShell Commands C-11

Telnet BeanShell Commands C-12

SNMP BeanShell Commands C-13

TLI BeanShell Commands C-14

PART 1

Overview of the Customization Tools

This part provides an overview of the tools available to customize Cisco Prime Network:

- Prime Network Customization Overview, page 1-1

CHAPTER 1

Prime Network Customization Overview

Prime Network provides the ability to customize the application in order to extend its functionality beyond what is provided "out-of-the-box".

Table 1-1 describes the extensions you can make to Prime Network and where to find the procedures that will guide you through the process.

Table 1-1 Supported Extensions for Prime Network

To make this extension:	Use these tools:	Described here:
Extend the Information Model		
Extend the set of supported properties for a network element and display them in the Inventory window.	Soft Properties	Modeling and Displaying Additional NE Properties, page 2-1
Add support for currently unsupported device types.	VNE Customization Builder	Adding Support for Additional Device Types by Creating U-VNEs, page 3-6
Add support for currently unsupported device software versions	VNE Customization Builder	Adding Support for Additional Software Versions, page 3-11
Add support for new standard and pluggable modules	VNE Customization Builder	Add Support for Additional Modules, page 3-12
Create New Commands and Activation Workflows for Configuring Devices		
Create new device commands and add them to the GUI	Command Manager Command Builder	Creating Commands and Command Scripts to Perform Device Configuration Operations, page 4-1
Create new transactions to allow users to deploy activations to devices	Transaction Manager	Deploying Activation Workflows to Devices Using Transaction Manager, page 5-1
Extend the GUI Client		
Add launch points to external applications	External Launch procedure	Adding External Launch Points to the GUI Client, page 8-1

Table 1-1 Supported Extensions for Prime Network (continued)

To make this extension:	Use these tools:	Described here:
Extend or Change Fault Management Capabilities		
Add support for currently unsupported events	VNE Customization Builder	Adding Support for New Events Using the VCB, page 6-1
Change the way Prime Network deals with events	VNE Customization Builder	Customizing Events, page 6-15
Create new threshold-crossing alarms	Soft Properties	Adding New Threshold-Crossing Alarms, page 7-1

Tools for Extending Prime Network

This topic provides a brief introduction to the tools you can use to extend the features of Prime Network.

Soft Properties

The Soft Properties tool in Prime Network Vision enables you to extend the set of supported properties for a network element (NE), by adding new (soft) properties. These properties extend the Cisco Information Model Object (IMO) and are available through the client GUI as well as through the Broadband Query Language (BQL) API. Soft properties are retrieved from the NE using Simple Network Management Protocol (SNMP) or Telnet/Secure Shell Protocol (SSH). In addition, alarm thresholds enable you to constantly monitor selected properties and generate an alarm every time these properties cross a user-defined threshold or violate a condition. The Soft Properties tool is launched by right-clicking a VNE and choosing **Management > Soft Properties Management**. To use this tool, you must have Configurator privileges.

Command Manager and Command Builder

Prime Network Command Manager and Command Builder allow you to create user-created commands and scripts that perform device configuration operations. Commands can range from simple show commands to wizards with multiple pages and input methods such as check boxes and drop-down lists. The commands you create execute a programmable sequence of SNMP or Telnet command lines. Once you have created and tested a new command, you can apply it to a wider scope of NEs. The new commands will be listed in the NE's context-sensitive menu and can be executed from there. The commands can also be used to create transactions which call a sequence of commands.

Command Manager uses the Command Builder back-end, but it provides the following additional features:

- View the command repository, which displays all device-level commands that are available in the system
- Execute commands on multiple devices
- Use input from files to provide parameter substitution
- Apply versions to individual commands
- Create command packages

However, Command Manager commands can only be applied at the network element (IManagedElement) level, while Command Builder commands can be associated with any existing object group (IMO), type, or instance.

You can launch Command Manager from the main menu of the Change and Configuration Management (CCM) GUI. You must have Configurator privileges to run Command Manager. To open CCM, use the following URL:

http://*gateway-IP*:8043/ccmweb/ccm/login.htm

To launch Command Builder, right-click a VNE and choose **Management**. This client also requires Configurator privileges.

Transaction Manager

Transaction Manager provides a GUI framework you can use to schedule and run transactions (activation workflows) that are created using the Prime Network XDE Eclipse SDK. It replaces the Workflow and Activation Wizard Builder features that were provided in previous releases of Prime Network. You can also run transactions using the NBI.

You can launch Transaction Manager from the main menu of the Change and Configuration Management (CCM) GUI. You must have Configurator privileges to run Transaction Manager. To open CCM, use the following URL:

http://*gateway-IP*:8043/ccmweb/ccm/login.htm

VNE Customization Builder (VCB)

The VCB is a tool that allows you to enable support for currently unsupported device types, software versions, and modules. You can also use the VCB to make Prime Network process unsupported syslogs, traps, or service events as Prime Network events, or to customize the way in which Prime Network processes events.

You must have Configurator or Administrator privileges to use the VCB. The VCB GUI client is launched from the Prime Network Administration GUI client by choosing **Tools > VNE Customization Builder** or by using the following URL:

https://*gateway-ip*:8043/prime-network-web/login.jsp

External Launch

You can add launch points to external applications or URLs to the right-click menu of NEs, links, tickets, and events. When the external launch point is clicked, it invokes the external application (for example, a script or batch file), or opens the specified URL in the default browser. You can also add filters so that the launch points are only displayed if the instance meets the filter requirements. You must have Configurator or higher privileges to use this procedure.

Editing the Registry

The Prime Network registry is a collection of xml files (called hives) that comprise and control the Prime Network system configuration. The registry contains almost all definitions and configurations used by Prime Network, including whether service alarms are correlated, the settings that control flapping, the severity, and so forth. You can change these parameters using the Registry Controller available from the Administration GUI client. Other parameters can be changed using the CLI. Those parameters are documented in this guide.

PART 2

Extending the Information Model

These topics explain how you can extend the Prime Network information model to support and display additional NE properties and to support additional device types, software versions, and modules:

- Modeling and Displaying Additional NE Properties, page 2-1
- Adding Support for New Devices, Software Versions and Modules Using the VCB, page 3-1

CHAPTER 2

Modeling and Displaying Additional NE Properties

Prime Network models and displays a fixed set of network element properties in the Inventory windows. If there are additional properties you want to display and monitor, you can do so using Soft Properties.

These topics explain how to use Soft Properties to add new network element properties to the Inventory window and control how these new properties are displayed:

- What are Soft Properties?, page 2-1
- Steps for Adding New NE Property Information to the Inventory, page 2-3
- Displaying a New Property in the Inventory Window, page 2-4
- Adding a New Inventory Table Containing SNMP get Details, page 2-11
- Making the Soft Property Available to Other NEs, page 2-14

Note We recommend that the users who manage Soft Properties have the following:

- For their user account—Configurator or higher default permissions.
- For all assigned device scopes—Configurator or higher security level.

What are Soft Properties?

Soft properties are definitions that contain rules for retrieving new NE information, parsing the retrieved information, and displaying it in the Inventory window, either as an additional property or as a table. In effect, soft properties can be used to extend the default subset of device properties that Prime Network models. Using soft properties allows you to quickly adapt to new software upgrades and new requirements that arise during ongoing operation and deployment.

For example, consider the case where the Prime Network system monitors the port parameters of an ATM switch, and the operator installs a new software version on the switch that is capable of reporting the bit error rate (BER) for each of the ports. Since this capability was not supported in previous software versions of the NE, the Prime Network VNE might not support the property. To avoid the need for a new VNE from Cisco, you can immediately support the new BER feature by creating a soft property.

A soft property is implemented through a set of definitions that determine how to retrieve, parse, and display a certain MIB variable from the NE. Information can be retrieved from the NE using SNMP or Telnet/SSH, and the definitions are stored in the XML registry. You define the protocol and rules using the Soft Properties GUI client. The new properties are applied in runtime and do not require system restart.

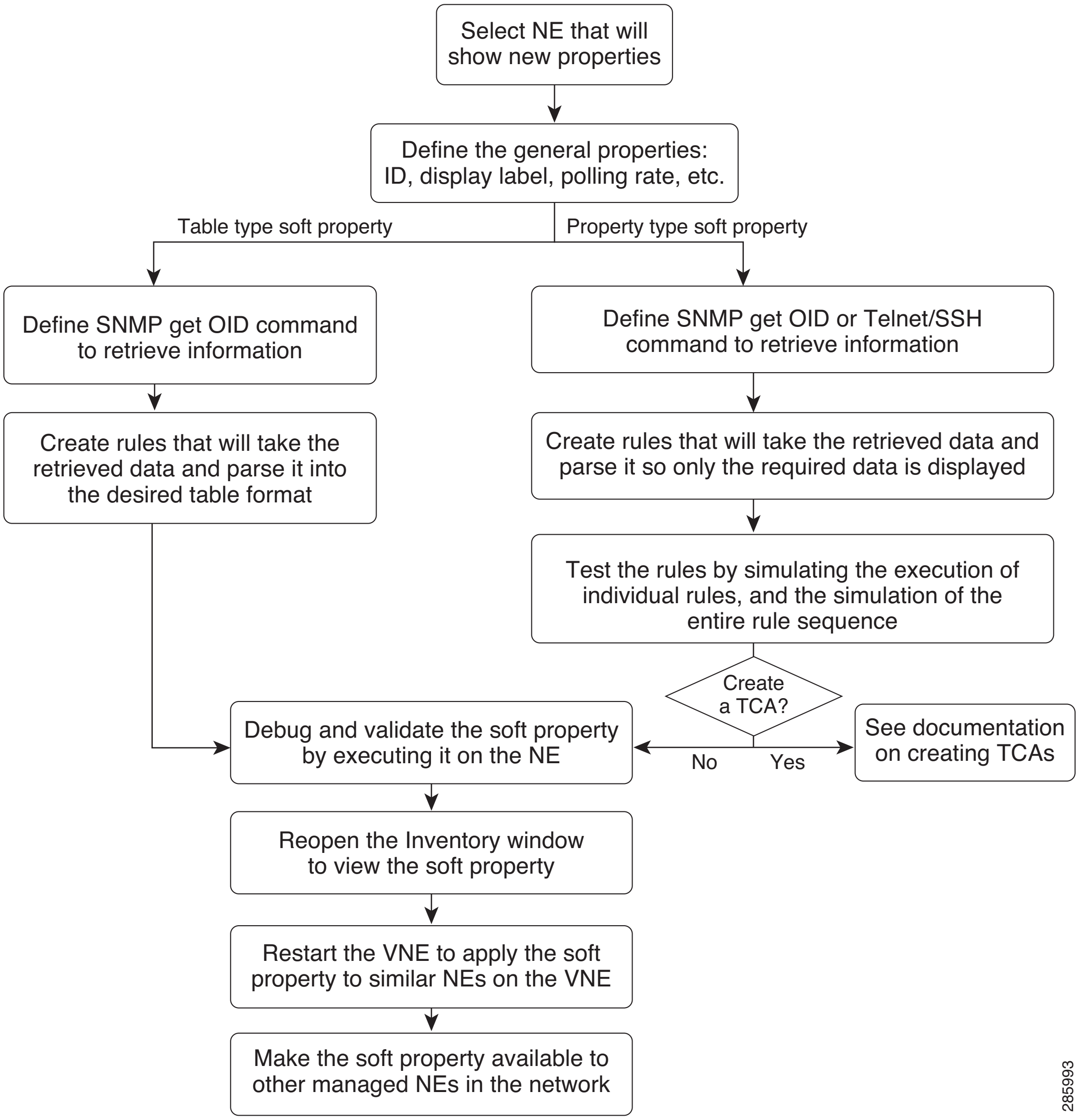
You can also associate alarm conditions with a soft property and create a new TCA that will be displayed in the Prime Network Vision and Events GUI clients. Creating TCAs is described in Creating a New Threshold-Crossing Alarm, page 7-1.

When you develop the soft property, only the specific NE instance is affected so that you can debug it and determine any impact it may have on the VNE. When you restart the VNE, the soft property is applied to all similar components on that VNE. Then you can apply it to multiple devices and components.

Steps for Adding New NE Property Information to the Inventory

Figure 2-1 illustrates the steps required to define a new soft property.

Figure 2-1 Flow for Creating New Soft Properties


Select NE that will show new properties
Define the general properties: ID, display label, polling rate, etc.
Table type soft property
Property type soft property
Define SNMP get OID command to retrieve information
Create rules that will take the retrieved data and parse it into the desired table format
Define SNMP get OID or Telnet/SSH command to retrieve information
Create rules that will take the retrieved data and parse it so only the required data is displayed
Test the rules by simulating the execution of individual rules, and the simulation of the entire rule sequence
Create a TCA?
No
Yes
See documentation on creating TCAs
Debug and validate the soft property by executing it on the NE
Reopen the Inventory window to view the soft property
Restart the VNE to apply the soft property to similar NEs on the VNE
Make the soft property available to other managed NEs in the network
285993

The steps are described in these topics:

- Displaying a New Property in the Inventory Window, page 2-4
- Adding a New Inventory Table Containing SNMP get Details, page 2-11
- Creating a New Threshold-Crossing Alarm, page 7-1

Displaying a New Property in the Inventory Window

This procedure leads you through the process of creating a new Property type so that new NE attributes can be modeled and displayed in the Inventory window. This allows you to monitor a new attribute on an ongoing basis.

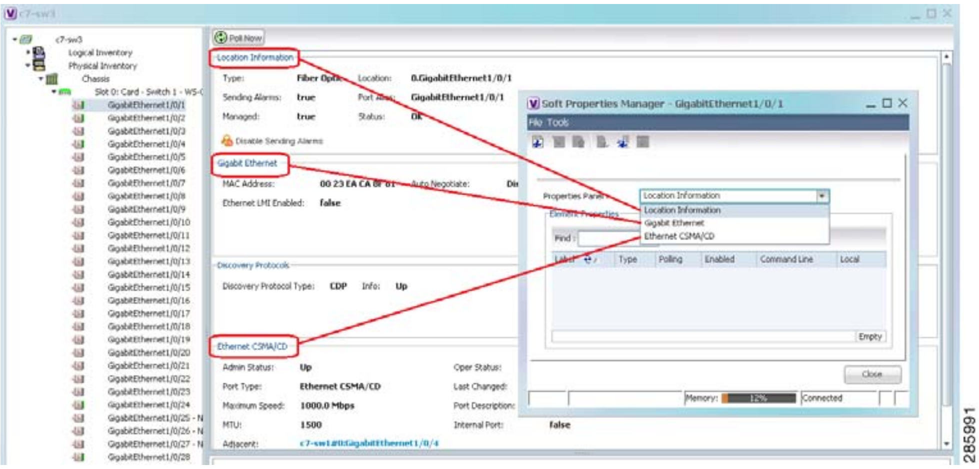
To create a new Property type soft property:

Step 1 In Prime Network Vision, right-click the node on which you want to apply a new soft property. The node can be the NE itself or an inventory node. For example, you can create a soft property on a port.

Step 2 Choose **Management > Soft Properties Management**.

The Soft Properties Manager dialog box opens and displays any existing soft properties for the selected node. As shown in Figure 2-2, the choices in the Properties Panel drop-down list align to the panels in the Inventory window for the selected node. For example, the properties panels for an ATM port are Location Information, ATM, and DS3.

Figure 2-2 Soft Properties Manager Dialog Box



Type: Fiber Optic Location: 0.GigabitEthernet1/0/1
Sending Alarms: true Port Alias: GigabitEthernet1/0/1
Managed: true Status: OK

MAC Address: 00 23 EA CA 0F 01 Auto Negotiate: Di
Ethernet LMI Enabled: false

Discovery Protocol Type: CDP Info: Up

Admin Status: Up Oper Status:
Port Type: Ethernet CSMA/CD Last Changed:
Maximum Speed: 1000.0 Mbps Port Description:
MTU: 1500 Internal Port: false
Adjacent: c7-sw1#0:GigabitEthernet1/0/4

Properties Panel: Location Information

Find :

Step 3 From the Panel Properties drop-down list, choose the inventory panel in which you want the new property to be displayed.

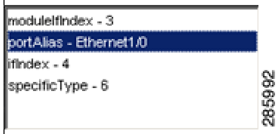
Step 4 Open the Add Soft Property dialog box by choosing **File > New Element**.

Step 5 In the General tab, define the general parameters for the soft property.

Field	Description	Example
Name	Enter a unique identifier for the soft property.	**sp01**
Label	Identifying text that will be displayed in Inventory window.	**My Soft Property**
Description	(Optional) Description of the soft property	**Example of soft property**
Type	Type of soft property, which determines the format of the information. In this case, select **Property** to display the defined information in the form of an additional field in the Inventory window. Telnet/SSH or SNMP is used to retrieve and parse information from the NE.	**Property**

Field	Description	Example
Polling Rate	Rate at which Prime Network will poll the NE for the information. Prime Network applies the appropriate default. Do not change this unless you are familiar with polling and the implications of choosing different rates.	**Status**
Enabled	Whether the VNE is running the command. This field allows you to temporarily disable soft properties, if needed.	[selected]

Step 6 In the Parsing tab, select the protocol you want to use to retrieve the information.

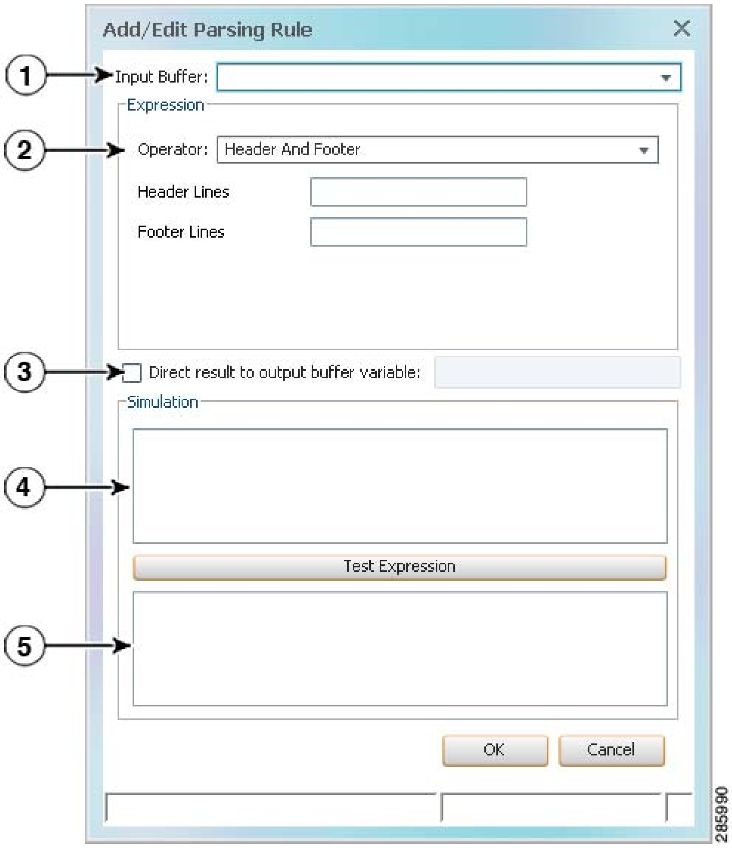
<table>
<tr><th>Protocol</th><th>Description</th></tr>
<tr><td>Use SNMP get(OID)</td><td>SNMP retrieval expression. The SNMP OID must start with a dot.
1. Enter the OID in the text box; for example:
<code>.1.3.6.1.2.1.2.2.1.16.</code>
2. With the cursor in the text box, click <b>Ctrl-Spacebar</b> to display the available fields and their current values.
3. Choose a field from the drop-down list and press <b>Enter</b>. The variable is populated in the text box with surrounding $ characters. This example retrieves the interface OutOctets:
<code>.1.3.6.1.2.1.2.2.1.16.$ifIndex$</code></td></tr>
<tr><td>Use Telnet/SSH</td><td>The Telnet/SSH retrieval expression. (To use Telnet, it must be enabled on the device.)
1. Enter the Telnet/SSH command in the text box; for example:
<code>show interface</code>
With the cursor in the text box, click <b>Ctrl-Spacebar</b> to display the available fields and their current values. For example:

moduleIfIndex - 3
portAlias - Ethernet1/0
ifIndex - 4
specificType - 6

2. Choose a field from the drop-down list and press <b>Enter</b>. The variable is populated in the text box with surrounding $ characters. For example, when this command is executed, <i>$portalias$</i> will be replaced with Ethernet1/0:
<code>.1.3.6.1.2.1.2.2.1.16.$portAlias$</code></td></tr>
</table>

Step 7 Define parsing rules. You can create multiple rules in order to retrieve exactly the information you require. Click **Add** to start defining your parsing rules.

The Add/Edit Parsing Rule dialog box appears.

Figure 2-3 *Add/Edit Parsing Rule Dialog Box*



Operator: Header And Footer

1	Input Buffer—Determines what input will be used for the parsing rule. The drop-down list in this field contains a blank item representing the default input, and a list of variables created by redirecting the output of previous rules (if any). If the input buffer is blank (default), the output of the previous parsing rule is used as this rule's input unless the output of the previous rule was directed to an output variable. In this case, the default input is the output of the last rule that was not directed to an output variable. For example, you create Rule 1. Then you create Rule 2 and Rule 3, but you redirect Rule 3's output to a variable. When you create Rule 4, it will use the output from Rule 2 as its input. That is because Rule 2 is the last rule with output that was *not* redirected.
2	Operator—Specifies a parsing operator. The operator you choose determines the fields that appear below it (Remove Lines, Match, Parse Integer, and so forth). See Supported Parsing Rules and Operators, page A-1.
3	Direct result to output buffer variable check box—When checked, you can save the parsing output as a variable with a unique name. This means that the output will not automatically be used as the input for the next rule, but the variable will be available for selection as the input for subsequent rules defined for this soft property.
4	Simulation panel—Allows you to test the rule by entering your rule and clicking **Test Expression**. This information is not saved when you close the window.
5	The parsed result that is displayed after clicking **Test Expression**. This information is not saved when you close the window.

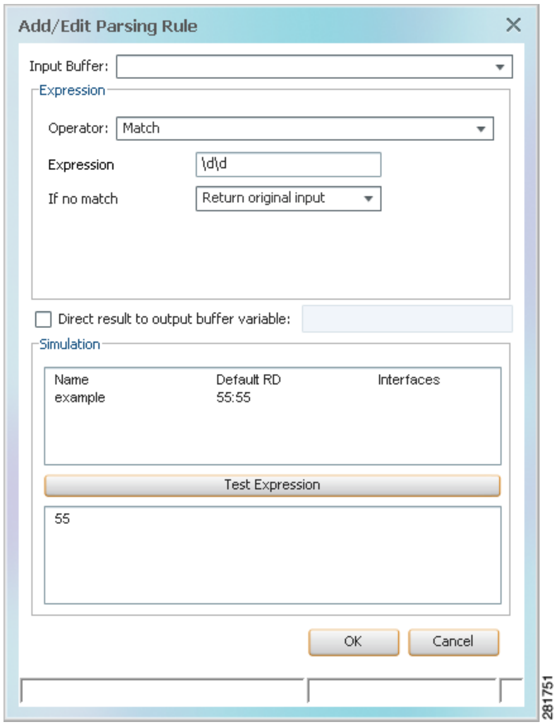
Step 8 To simulate the parsing rule without sending the command to the device:

a. Enter the text into the top Simulation window. For example:

```
Name              Default RD             Interfaces
example            55:55
```

b. Click **Test Expression**. The parsing rule result is displayed, as shown in Figure 2-4.

Figure 2-4 Match Parsing Rule



Add/Edit Parsing Rule

Input Buffer:

Expression

Operator: Match

Expression \d\d

If no match Return original input

□ Direct result to output buffer variable:

Simulation

```
Name              Default RD             Interfaces
example           55:55
```

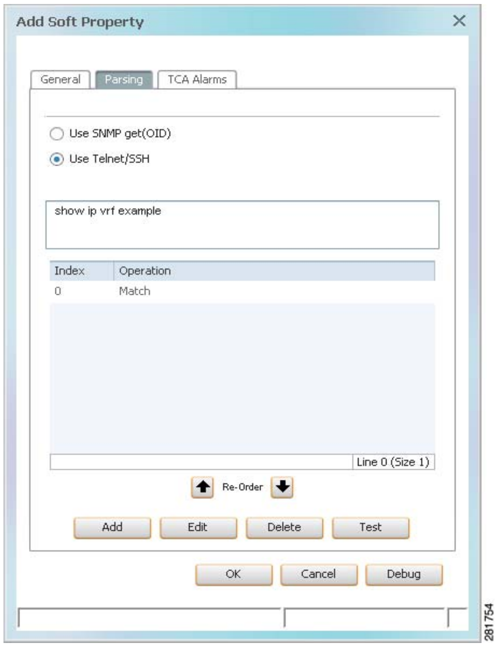
Test Expression

```
55
```

OK Cancel

Step 9 Click **OK**. The rule is added to the Parsing tab, as shown in Figure 2-5.

Figure 2-5 Parsing Tab with Match Parsing Rule



○ Use SNMP get(OID)
● Use Telnet/SSH

show ip vrf example

Index	Operation
0	Match

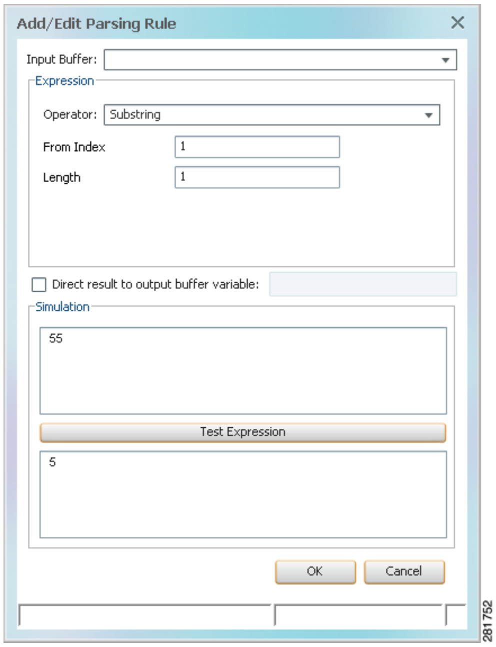
Line 0 (Size 1)

Step 10 Click **Add** to create another parsing rule. For example:

Operator	**Substring**
From Index	**1**
Length	**1**
Simulation text box	**55**

Step 11 Click **Test Expression**. The result 5 is displayed in the Result text box, as shown in Figure 2-6.

Figure 2-6 Substring Parsing Rule



Input Buffer:

Operator: Substring

From Index: 1

Length: 1

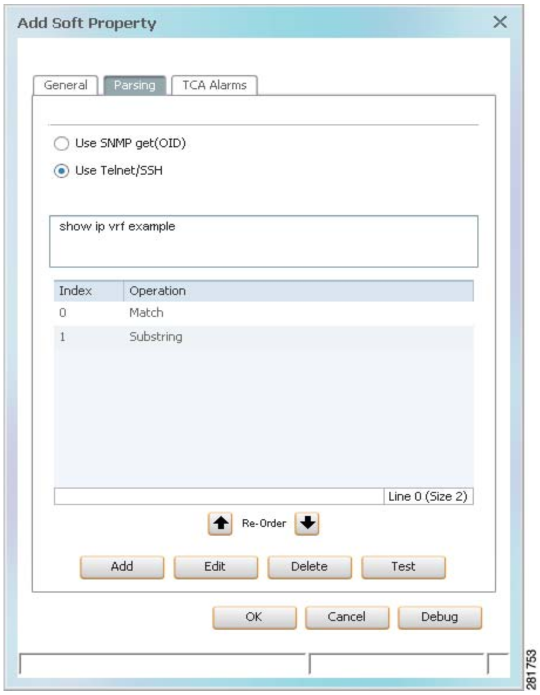
□ Direct result to output buffer variable:

55

5

Step 12 Click **OK**. The rule is added to the Parsing tab, as shown in Figure 2-7.

Figure 2-7 Parsing Tab with Match and Substring Parsing Rules



○ Use SNMP get(OID)

● Use Telnet/SSH

show ip vrf example

Index	Operation
0	Match
1	Substring

Line 0 (Size 2)

Step 13 To simulate the rule sequence without sending any commands to the device:

a. From the Add Soft Property dialog box (Parsing tab), click **Test** to open the Test Parsing Rules dialog box.

b. Copy the output of the native commands into the Input Buffer.

c. Click Test (in the Test Parsing Rules dialog) and verify the results.

a. Click **Close**.

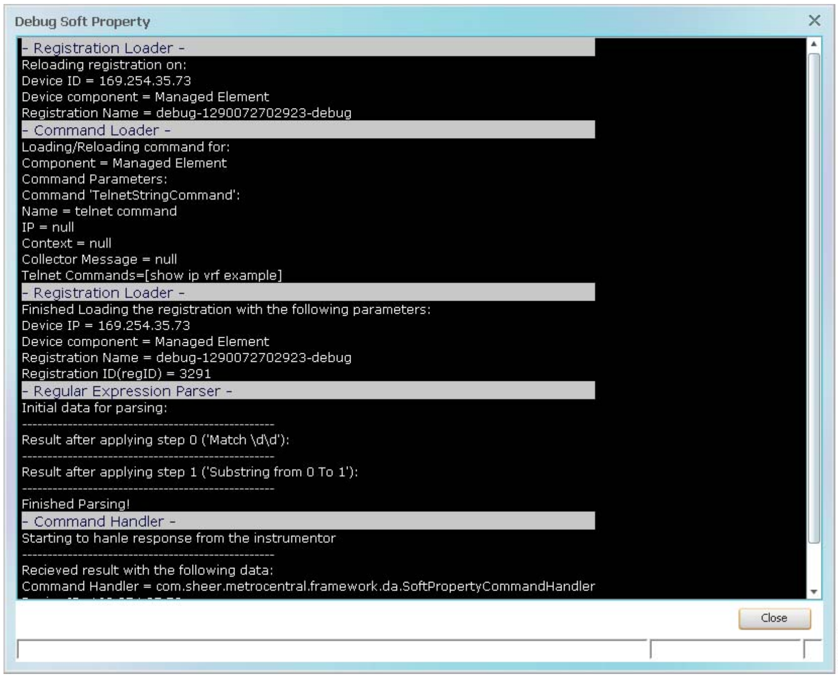
Step 14 To send the command to the device to check its validity, in the Add Soft Properties dialog box, click **Debug**.

Note No notification message is displayed when a VNE fails to collect properties. The properties are simply not populated in the GUI.

The results of the test are displayed Debug Soft Property dialog box.

Step 15 The Debug Soft Property dialog box is displayed.

Figure 2-8 *Debug Soft Property Dialog Box*



```
- Registration Loader -
Reloading registration on:
Device ID = 169.254.35.73
Device component = Managed Element
Registration Name = debug-1290072702923-debug
- Command Loader -
Loading/Reloading command for:
Component = Managed Element
Command Parameters:
Command 'TelnetStringCommand':
Name = telnet command
IP = null
Context = null
Collector Message = null
Telnet Commands=[show ip vrf example]
- Registration Loader -
Finished Loading the registration with the following parameters:
Device IP = 169.254.35.73
Device component = Managed Element
Registration Name = debug-1290072702923-debug
Registration ID(regID) = 3291
- Regular Expression Parser -
Initial data for parsing:
-----------------------------------------------
Result after applying step 0 ('Match \d\d'):
-----------------------------------------------
Result after applying step 1 ('Substring from 0 To 1'):
-----------------------------------------------
Finished Parsing!
- Command Handler -
Starting to hanle response from the instrumentor
-----------------------------------------------
Recieved result with the following data:
Command Handler = com.sheer.metrocentral.framework.da.SoftPropertyCommandHandler
```

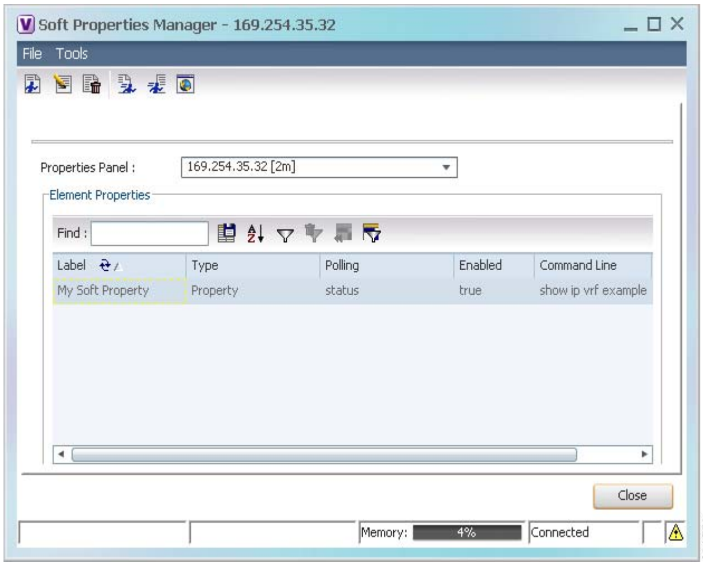
Step 16 After the Status is returned as valid, confirm that the input parameters have returned the correct values; for example:

```
Telnet Commands=[show ip vrf example]
```

Step 17 Click **Close** to exit the Debug Soft Property window.

Step 18 Click **OK** to exit the Add Soft Property dialog box. The new soft property is listed in the Soft Properties Manager dialog box, as shown in Figure 2-9.

Figure 2-9 Soft Properties Manager Window



Properties Panel : 169.254.35.32 [2m]

Element Properties

Find :

Label	Type	Polling	Enabled	Command Line
My Soft Property	Property	status	true	show ip vrf example

Step 19 Click **Close** to exit the Soft Properties Manager.

The soft property is activated on the NE. You should see the new information in your Inventory window the next time you open it. When you restart the VNE, the soft property is applied to all similar components on the VNE.

Take the appropriate next steps:

- To add a TCA to the soft property, see Creating a New Threshold-Crossing Alarm, page 7-1.
- To make other NEs display the property, see Making the Soft Property Available to Other NEs, page 2-14.

Adding a New Inventory Table Containing SNMP get Details

This procedure leads you through the process of creating a new Table type soft property that extracts information from the MIB and displays it (in a table) in the Inventory window.

To create a new Table type soft property:

Step 1 In Prime Network Vision, right-click the node on which you want to apply a new soft property. The node can be the NE itself or an inventory node. For example, you can create a soft property on a port.

Step 2 Choose **Management > Soft Properties Management**.

The Soft Properties Manager dialog box opens and displays any existing soft properties for the selected node. As shown in Figure 2-2, the choices in the Properties Panel drop-down list align to the panels in the Inventory window for the selected node. For example, the properties panels for an ATM port are Location Information, ATM, and DS3.

Step 3 From the Panel Properties drop-down list, choose NE inventory panel in which you want the new property table to be displayed.

Step 4 Open the Add Soft Property dialog box by choosing **File > New Element**.

Step 5 Define the general parameters for the soft property in the General tab.

Field	Description	Example
Name	Enter a unique identifier for the soft property.	**sp02**
Label	Identifying text that will be displayed in Inventory window.	**My Soft Table**
Description	(Optional) Description of soft property	**Example of a Soft Table**
Type	Type of soft property, which determines the format of the information. In this case, select **Table** to display the defined information in tabular format. An SNMP **get** is performed to retrieve NE information and display it in tabular form in the Inventory window.	**Table**
Polling Rate	Rate at which Prime Network will poll the NE for the information. Prime Network applies the appropriate default. Do not change this unless you are familiar with polling and the implications of choosing different rates.	**Status**
Enabled	Whether the VNE is running the command. This field allows you to temporarily disable soft properties, if needed.	(Selected)

Step 6 Define the SNMP retrieval expression. The SNMP OID must start with a dot.

a. Enter the OID in the text box; for example:

```
.1.3.6.1.2.1.4.20.1.
```

b. With the cursor in the text box, click **Ctrl-Spacebar** to display the available fields and their current values.

c. Choose a field from the drop-down list and press **Enter**.

Step 7 Click **Add** and define the column and data information in the Add Edit Column Controller dialog box. For example:

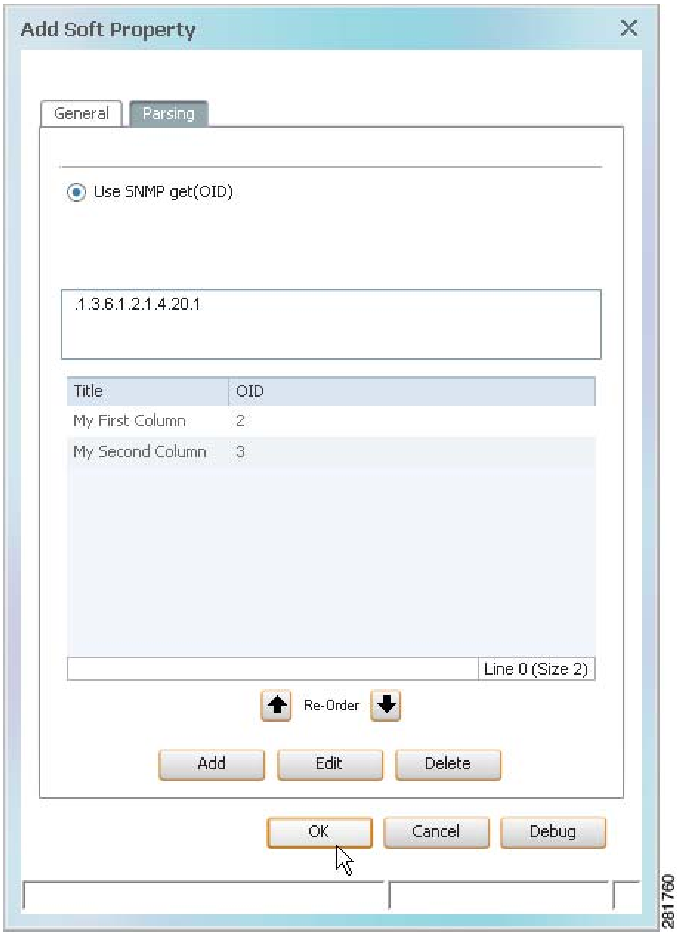
Column Title	**My First Column**
Column Data	**2**

Step 8 Add more column titles and data as needed; for example:

Column Title	**My Second Column**
Column Data	**5**

When you are done, click **OK**. See the example in Figure 2-10.

Figure 2-10 Add Soft Property Dialog Box



● Use SNMP get(OID)

.1.3.6.1.2.1.4.20.1

Title	OID
My First Column	2
My Second Column	3

Line 0 (Size 2)

Step 9 To send the command to the device to check its validity, in the Add Soft Properties dialog box, click **Debug**.

Note No notification message is displayed when a VNE fails to collect properties. The properties are simply not populated in the GUI.

When you finish testing the rules, click **Close** to close the Debug Soft Property dialog box.

Step 10 Click **OK** to exit the Add Soft Property dialog box.

Step 11 Click **Close** to exit the Soft Properties Manager dialog box.

Step 12 Click **Close** to exit the Soft Properties Manager.

The soft property is activated on the NE. You should see the new information in your Inventory window the next time you open it. When you restart the VNE, the soft property is applied to all similar components on the VNE.

To make other NEs display the property, see Making the Soft Property Available to Other NEs, page 2-14.

Making the Soft Property Available to Other NEs

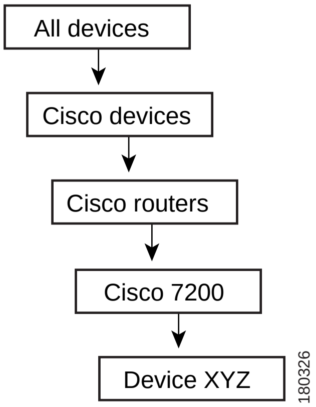
A property definition is applicable to all objects of the same type in the selected NE. After the soft property has been defined and tested on a specific instance of a managed element, it can be *published* and applied to a wider scope of managed NEs in the network. This requires moving the property definition from the specific NE instance to a higher level in the registry hierarchy.

The Soft Properties Publish Controller dialog box enables you to publish the soft property to one or more locations across the inheritance hierarchy (as defined in the system). In other words, you define the scope where the soft property is to be applied in the hierarchy.

Note Users with Configurator privileges can add and publish soft properties on all devices.

Figure 2-11 shows an example of the inheritance hierarchy. In this example, the top level of the hierarchy is All devices, and the lowest level of the hierarchy is Device XYZ.

Figure 2-11 Inheritance Hierarchy Example



When a soft property is published to a node in the hierarchy, it overrides any inherited soft properties from a higher level and applies automatically to all its children. For example, if a soft property is published to Cisco 7200 devices, it overrides any variant of this soft property which is defined at a higher level, and is assigned to all devices of type Cisco 7200 in the system.

Caution Soft property publishing can affect system memory usage, device utilization, and system performance. For example, publishing a property that contains the **show running-config** command to many devices could negatively impact system memory usage and device utilization. We strongly recommend that you measure the effect of publishing the soft property on system memory usage before you publish the soft property. To view the changes, you must restart the VNE or unit for the publishing to take effect.

To publish a soft property:

Step 1 In Prime Network Vision, right-click a managed element, then choose **Management > Soft Properties Management**. This opens the Soft Properties Manager window.

Step 2 Select the soft property you want to publish. Remember that the soft properties that are displayed depend on the NE, and the choice you make in the Properties Panel drop-down list.

Step 3 Open the hierarchy manager by choosing **Tools > Hierarchy Manager**.

Each row that is displayed in the hierarchy manager table represents a different level of the hierarchy. The rows are displayed in descending order; the top row is the highest level of the hierarchy and the bottom row is the lowest level of the hierarchy.

The following information is displayed in the table:

- Exist—When a node in the hierarchy is selected, it indicates that a local variant of the soft property exists for that node.
- VNE Hierarchy Location—The hierarchy path, as defined in the registry.
- IMO Class Name—Currently unavailable in this version.

Table 2-1 describes the icons that appear.

Table 2-1 Hierarchy Manager Window Tools

Icon	Name	Description
⧉	Copy	Copies the soft property from a selected node in the hierarchy to another node in the hierarchy. A copy icon is displayed to the left of the selected node.
✂	Cut	Cuts the soft property from a selected node in the hierarchy to move it to another node in the hierarchy. A cut icon is displayed to the left of the selected node.
📋	Paste	Pastes the soft property that was copied or cut from a selected node in the hierarchy to another node in the hierarchy. A paste icon is displayed to the left of the selected node.
🗑	Delete	Deletes the soft property from the selected node in the hierarchy. **Note** If the soft property has been deleted from all nodes, it is removed from the list in the main dialog box of the Soft Properties Manager.

Step 4 Select the node in the hierarchy from which you want to publish the soft property.

Step 5 Click the Copy or Cut icons to perform the required action.

Step 6 Select the node in the hierarchy where you want to publish the soft property, and click the Paste icon.

Step 7 Restart the VNE on which you want to publishing to take effect.

Adding Support for New Devices, Software Versions and Modules Using the VCB

This chapter describes using the VNE Customization Builder (VCB) to add support for currently unsupported devices, software versions, and modules. For information about adding support for new events, see Adding Support for New Events Using the VCB, page 6-1.

This chapter contains the following topics:

- VCB Overview, page 3-1
- Comparison of Generic SNMP VNEs, U-VNEs, and Developed VNEs, page 3-3
- Adding Support for Additional Device Types by Creating U-VNEs, page 3-6
- Adding Support for Additional Software Versions, page 3-11
- Add Support for Additional Modules, page 3-12
- Testing and Certifying VCB Customizations, page 3-14
- Deploying VCB Customizations to Your Production Environment, page 3-25
- Deleting VNE Customizations from the Registry, page 3-28

VCB Overview

The VCB is a tool that allows advanced users to extend the "out-of-the-box" support and capabilities of Prime Network:

- Enable discovery of currently unsupported device types by creating user-defined VNE drivers, known as U-VNEs.
- Clone from an existing VNE driver to manage new devices that belong to an existing supported device family.
- Extend the discovery and management capabilities of existing VNE drivers to enable Prime Network to:
  - Recognize cards that would otherwise be treated as "Unknown".
  - Process syslogs, traps, or service events as Prime Network events.
  - Recognize additional software versions, such as maintenance releases of Cisco IOS and other software.
- Produce a list of unsupported traps for a particular MIB and add them as Prime Network events.

Note An event is generated for all VCB write operations (add, update, override, and delete) and is displayed in the System tab in the Prime Network Events GUI client.

VCB GUI vs. VCB CLI

VCB functions can be performed using the GUI interface that is available in the Prime Network Administration application (from the Tools menu), or by executing VCB commands in the Command Line Interface (CLI).

If you are new to the VCB, we recommend using the VCB GUI. If you are already familiar with using the CLI to perform VCB functions, or you want to perform more advanced VCB functions that are not yet included in the GUI, you can use the CLI.

Accessing the VCB

You can access the VCB through Prime Network Administration or by specifying a URL in your web browser. To open the VCB:

Step 1 In Prime Network Administration, choose **Tools** > **VNE Customization Builder**.

or

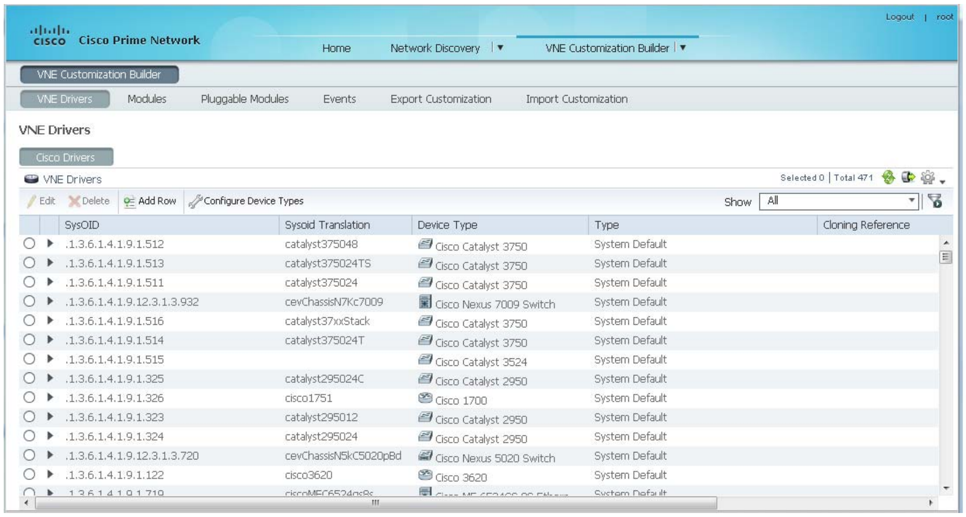
In your web browser, enter the following URL. The *gateway-IP* can be an IPv6 address.

https://*gateway-IP*:8043/prime-network-web

Step 2 Enter the login credentials and then select **VNE Customization Builder** in the Home page. The VCB opens, and the VNE Drivers page is displayed.

Note Only two users can be logged into the VCB GUI client at one time.

Figure 3-1 VCB - VNE Drivers Page



		SysOID	Sysoid Translation	Device Type	Type	Cloning Reference
○	▶	.1.3.6.1.4.1.9.1.512	catalyst375048	Cisco Catalyst 3750	System Default	
○	▶	.1.3.6.1.4.1.9.1.513	catalyst375024TS	Cisco Catalyst 3750	System Default	
○	▶	.1.3.6.1.4.1.9.1.511	catalyst375024	Cisco Catalyst 3750	System Default	
○	▶	.1.3.6.1.4.1.9.12.3.1.3.932	cevChassisN7Kc7009	Cisco Nexus 7009 Switch	System Default	
○	▶	.1.3.6.1.4.1.9.1.516	catalyst37xxStack	Cisco Catalyst 3750	System Default	
○	▶	.1.3.6.1.4.1.9.1.514	catalyst375024T	Cisco Catalyst 3750	System Default	
○	▶	.1.3.6.1.4.1.9.1.515		Cisco Catalyst 3524	System Default	
○	▶	.1.3.6.1.4.1.9.1.325	catalyst295024C	Cisco Catalyst 2950	System Default	
○	▶	.1.3.6.1.4.1.9.1.326	cisco1751	Cisco 1700	System Default	
○	▶	.1.3.6.1.4.1.9.1.323	catalyst295012	Cisco Catalyst 2950	System Default	
○	▶	.1.3.6.1.4.1.9.1.324	catalyst295024	Cisco Catalyst 2950	System Default	
○	▶	.1.3.6.1.4.1.9.12.3.1.3.720	cevChassisN5kC5020pBd	Cisco Nexus 5020 Switch	System Default	
○	▶	.1.3.6.1.4.1.9.1.122	cisco3620	Cisco 3620	System Default	
○	▶	.1.3.6.1.4.1.9.1.719	ciscoMEC6524gs8s	[illegible]	System Default	

Note You can reorder the columns by dragging and dropping the column headers. You can also determine which columns will be displayed by using the Settings tool in the toolbar above the table.

Cisco vs. Non-Cisco Devices

The VCB displays the details of the Cisco and the third-party (non-Cisco) VNE drivers in separate tabs as Prime Network supports separate installation directories and registry service for Cisco and non-Cisco drivers. Non-Cisco VNEs do not support pluggable module specification, and the pluggable module information is retrieved from the network element itself.

Note The Non-Cisco Drivers tab in VCB GUI is displayed only after successful installation of a non-Cisco device package in Prime Network.

Comparison of Generic SNMP VNEs, U-VNEs, and Developed VNEs

Before using the VCB to add support for unsupported network elements, you should understand the difference between the types of VNEs.

Developed VNEs

A VNE for a supported NE that was developed by Cisco and supplied with the product or through a downloadable Device Package. These VNEs are created using the Prime Network Administration GUI client.

Note You should always use a developed VNE, if possible, and extend it using the VCB, if necessary. Developed VNE drivers are designed and tuned for specific technologies and devices, while U-VNEs model a subset of device capabilities.

Generic SNMP VNEs

Generic SNMP VNEs are usually created to monitor the most basic inventory information for an unsupported NE. These VNEs are created using the Prime Network Administration GUI client and by choosing Generic SNMP as the VNE Type. (Supported VNE types are AutoDetect, Cloud, ICMP, and Generic SNMP.)

Use the VCB to extend Generic SNMP VNEs to monitor unsupported events.

U-VNEs

A U-VNE is a user-defined VNE created using the VCB to enable support for an unsupported device. A U-VNE can be created from scratch, based on a specific U-VNE template, or it can be created by cloning an existing supported device.

Features, advantages, and limitations of template-based U-VNEs are template-dependent. The GenericUVNE template supplies the same level of modelling as a Generic SNMP VNE. You can then further extend the U-VNE for additional event recognition using the VCB.

Table 3-1 shows the differences levels of support for Prime Network features in Generic SNMP VNEs, U-VNEs, and developed VNEs.

Note The information in this table is for comparative purposes only. The support provided by each U-VNE template might vary from template to template.

Table 3-1 Features Supported in Generic SNMP VNEs, U-VNEs, and Developed VNEs

Feature	Generic SNMP VNEs	U-VNEs	Developed VNEs
Fault Analysis	Standard traps that are supported by Generic SNMP VNEs. Can be extended to support proprietary traps and syslogs.	Standard and proprietary traps and syslogs defined in the U-VNE template or NE instrumentation of cloned device. Syslogs are not supported by the GenericUVNE template but it can be extended to support proprietary traps and syslogs.	Standard and proprietary traps and syslogs.
MIB support	Standard MIB-II	Standard MIB-II and private MIBs for physical inventory discovery depending on template or cloned device. Can import traps to add event recognition. Based on most commonly used subset of standard MIB-II MIBs which some device manufacturers might alter; information from the proprietary implementation of third-party device vendors is not available through the use of standard MIBs.	Standard MIB-II and private MIBs
CLI or XML over Telnet is used to query NE	No	Protocol used to query depends on setting in template or cloned device.	Yes
Multivendor	Yes	Yes	Yes
NBI	Yes	Yes	Yes
Discovery	Yes	Yes (CDP is supported for Cisco devices). Template-based U-VNEs support the instrumentation defined in template	Yes
Physical Inventory, Containment	No	Yes, depending on setting in template or cloned device.	Yes, depending on the NE instrumentation
Logical Inventory and Technologies	Routing table, ARP table, default bridge, IP interfaces	Routing table, ARP table, default bridge, IP interfaces[1]	Technologies and NE instrumentation supported in Prime Network

Table 3-1 Features Supported in Generic SNMP VNEs, U-VNEs, and Developed VNEs (continued)

Feature	Generic SNMP VNEs	U-VNEs	Developed VNEs
Alarm Correlation (root cause analysis)	No	Yes (for factory-defined events only, provided that the related technologies are supported by the U-VNE[2] or cloned device). Events are associated to managed element. For events that were added via the VCB, correlation is done based on the DC key (correlation using network flow is not supported). Event is correlated to service events with the same source. Users cannot customize correlation parameters such as weight, correlation delay, and so on. Root-cause events cannot be added via the VCB.	Yes (for factory-defined events). For events that were added via the VCB, correlation is done based on the DC key (correlation using network flow is not supported). Event is correlated to service events with the same source. Users cannot customize correlation parameters such as weight, correlation delay, and so on. Root-cause events cannot be added via the VCB.
NE Image Management	No	Yes, if supported in the VNE from which it was cloned.	Yes
Configuration Archival	No	No	Yes
Path Tracing	No	Only physical and Ethernet are supported. Template-based U-VNEs: Provide limited support for path tracing. If U-VNE does not support routing, network paths that traverse the U-VNE will stop and the result of the trace will show only the path to or from the U-VNE. Exact behavior depends on technologies used by NE, level of adherence to standard MIB support, and location of U-VNE on the path.	Yes
Topology Discovery	Physical and Ethernet	Template-based U-VNEs: Only physical and Ethernet are supported. Supported technologies are documented in the templates. Dynamic topology discovery is limited to U-VNEs on the network edge that are connected to a developed VNE. U-VNEs support IP topologies on high-level data link control (HDLC) and serial ports in point-to-point links, with no duplicate IP addresses, and same IP subnet. Cloned U-VNEs: Depends on the VNE from which it was cloned.[1]	Full

[1] For a U-VNE driver cloned from a developed VNE driver, the logical inventory, technology and topology discovery instrumentation usage are inherited from the developed VNE driver.

[2] For example, link up/down correlates to the card-out service alarm, while MPLS TE Tunnel down is associated to managed element and is not correlated to other alarms.

Identifying Supported Devices

The VNE Drivers tab provides a tabular view of all the supported device types and identifies whether they are system-defined (native Prime Network support) or user-defined (support added by Prime Network user using the VCB). It also provides the following information for each supported device:

- SysOID—The sysOID of the device.
- SysOID Translation—the MIB name of the device, derived from the sysOID. This a unique identifier for the device and provides more granular identification of the device than the device type. For example, the device type might be Cisco 3750 but the SysOID identifies the exact model of 3750 that is supported, for example, Catalyst375024TS.
- Device Type—The device family to which the supported device belongs.
- Type—System-defined or user-defined.
- Cloning Reference—If the U-VNE was created by cloning, this shows the cloning method used, either GenericUVNE or Entitymib.
- Overriding System Default—Shows as True if a system-defined device has been extended.

To get more details for each supported device, click the arrow on the extreme left to expand the display.

Adding Support for Additional Device Types by Creating U-VNEs

Using the VCB, you can enable Prime Network to discover and manage devices that have no system-defined VNE driver and are therefore not currently supported. To enable support for additional device types, you create a U-VNE and can then add the VNE to Prime Network in the normal manner, using the Administration GUI client. It will be modeled and added to the Prime Network device inventory. The level of modeling depends on the amount of detail provided to the system when you create the U-VNE.

Prime Network creates the registry information for these VNEs and saves it in site.xml, which is the registry location where all local changes should be stored. As with all VNEs, to maintain a live model of each network element and the entire network, VNEs must have connectivity with the device.

Note User-defined VNEs created with the VCB are not a replacement for developed VNE drivers. Developed VNE drivers are packaged with the Prime Network product and are also available in downloadable Device Packages. Developed VNE drivers are designed and tuned for specific technologies and devices, while U-VNEs model a subset of device capabilities.

These topics describe the methods for adding support for a new device type:

- Methods for Creating U-VNEs—Overview, page 3-7
- Creating a U-VNE Using the GenericUVNE Template, page 3-7
- Create a U-VNE by Cloning, page 3-8

Methods for Creating U-VNEs—Overview

Several approaches are available for creating U-VNEs. Depending on the approach you choose, the modeling capabilities will be more specific or less specific. Table 3-2 lists the supported methods and how to choose the one that is appropriate for your situation.

Table 3-2 Methods for Creating U-VNEs for New Device Types

If the unsupported device...	Use this Method:	Described in:
Belongs to a supported *device series or device family*	Clone a developed VNE driver from the supported device series or family. An unsupported device in a supported series or family has the same or a very similar management interface as the developed VNE driver. The cloned U-VNE inherits the behavior of the source VNE driver, and can be further extended using the VCB to add more device module discovery and event recognition. This method produces the most specific modelling results.	Create a U-VNE by Cloning, page 3-8
Uses the same NE software version as another supported device	Clone a developed VNE driver that uses the supported software version. An unsupported device running a supported NE software version should share the same management interface as a supported device. The cloned U-VNE inherits the behavior of the source VNE driver, and can be further extended using the VCB to add more device module discovery and event recognition.	Create a U-VNE by Cloning, page 3-8
Does not belong to a supported device series or family, or does not run a supported NE software version	Create a U-VNE “from scratch” using the GenericUVNE template. This is typically the case for non-Cisco devices and Cisco device families that are not supported by Prime Network, where the instrumentation or management interface for the unsupported device does not match that of any supported VNE. **Note** Network core devices should not use GenericUVNE template. This method produces the least specific device modelling results.	Creating a U-VNE Using the GenericUVNE Template, page 3-7

Creating a U-VNE Using the GenericUVNE Template

The following example shows how to enable support for an unsupported device by creating a U-VNE. For the purposes of this example, we will create a U-VNE to enable support for a Linux server. The U-VNE will be based on the GenericUVNE template. See U-VNE Templates, page B-56 for more information about this template.

Before You Begin

- Make sure this is the proper method for your situation by checking the information in Methods for Creating U-VNEs—Overview, page 3-7.
- Obtain the sysOID of the NE you want to manage. The sysOID can be retrieved from the properties of the system event generated for an unsupported device (shown in Prime Network Vision or Prime Network Events). Alternatively, you can query the device using the snmp get command.

- Confirm that Prime Network does not support the sysOID. You can do this by filtering the VNE drivers list in the VCB GUI by the sysOID. If it is not found, it is not supported.

To create a U-VNE:

Step 1 Access the VCB, as described in Accessing the VCB, page 3-2.

Step 2 In the VNE Drivers tab, click **Add Row**.

Step 3 In the SysOID field, enter the sysOID of the new device. Note that after you have saved the U-VNE, the SysOID will be automatically translated to a unique device name plus identifier (if a compiled MIB version is available). This value will be displayed in the SysOID Translation column.

Step 4 Specify the name that will represent the device in the Prime Network GUI by creating a new device type, as follows:

a. In the Device Type field, click the down arrow to display the Device Type Selector.

b. Click the Tools icon in the upper right area of the Device Type Selector and select **Add**.

c. In the Device Type field, enter the device name that will represent the device in the Prime Network maps and element tables. We recommend that you prefix the device type name with the network element vendor name for easy identification.

d. In the Category field, select the category to which the device belongs. In this case, select **Server**.

e. In the Device Series field, specify the device series to which the device belongs.

f. Click **OK**. The new Device Type name will appear in the Device Type field.

Step 5 In the Type field, select **User Defined VNE - by Template**.

Step 6 In the Cloning Reference field, select **GenericUVNE**.

Step 7 Click **Save**.

Step 8 Complete tests and certify the customization; see Testing and Certifying VCB Customizations, page 3-14.

Create a U-VNE by Cloning

This section describes how to enable support for an unsupported device by cloning a developed VNE driver from a supported device series or family. The cloned U-VNE inherits the behavior of the source VNE driver, and can be further extended using the VCB to add more device module discovery and event recognition.

Before You Begin

- Make sure this is the proper method for your situation by checking the information in Methods for Creating U-VNEs—Overview, page 3-7.
- Obtain the sysOID of the NE you want to manage. The sysOID can be retrieved from the properties of the system event generated for an unsupported device (shown in Prime Network Vision or Prime Network Events). Alternatively, you can query the device using the snmp get command.
- Confirm that Prime Network does not support the sysOID. You can do this by filtering the VNE drivers list in the VCB GUI by the sysOID. If it is not found, it is not supported.

To create a U-VNE by cloning:

Step 1 Access the VCB, as described in Accessing the VCB, page 3-2.

Step 2 In the VNE Drivers tab, click **Add Row**.

Step 3 In the SysOID field, enter the sysOID of the new device.

Step 4 In the Device Type field, select the device type that will represent the new supported device in the Prime Network maps and element tables. If necessary, you can create a new device type from the Tools icon in the Device Type selector.

Step 5 To clone from a supported device family: In the Type field, select **User Defined VNE - by Device Family**.
To clone from a supported software version: In the Type field, select **User Defined VNE - by Software Version**, then select the device series and the scheme.

Note Cloning by software version performs two customizations in a single operation. It clones both the device type and software versions. The new driver will be listed in the GUI as cloned by device.

Step 6 In the Cloning Reference field, specify the item from which you want to clone the new VNE.

Step 7 Click **Save**.

Step 8 Complete tests and certify the customization; see Testing and Certifying VCB Customizations, page 3-14.

Defining a Device Type to Determine the GUI Representation of VNE Drivers

Each VNE is represented by a specific name and icon in the maps and element tables. The device type determines the name used to represent the VNE while the device family determines the icon used to represent the VNE. When you create a U-VNE, you need to select a device type so that the system knows how to represent the new device in the GUI. If none of the existing device types is a suitable representation of your U-VNE, you can create a new device type. You can create a new device type in one of the following ways:

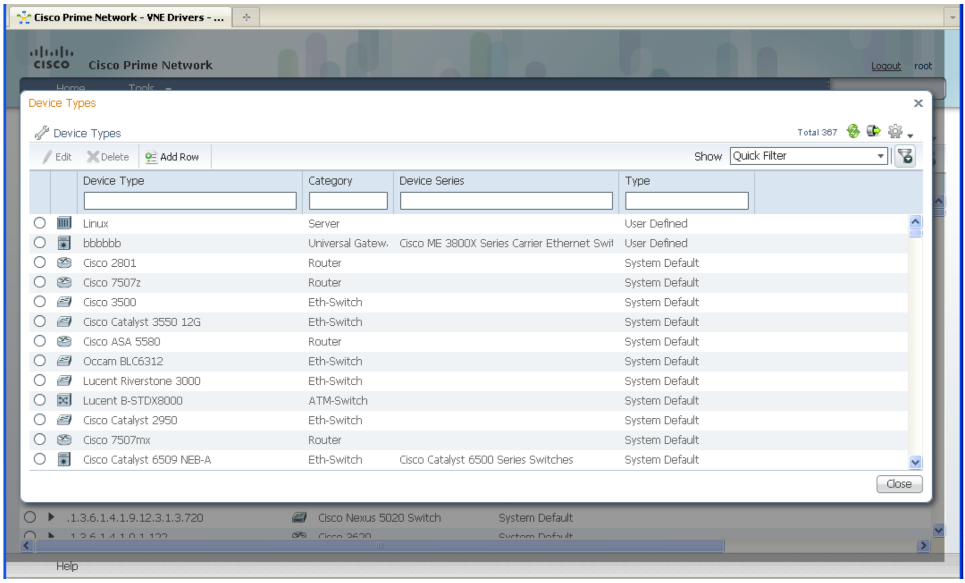
- From the VNE drivers window, during the VNE driver creation/editing process. See Creating a U-VNE Using the GenericUVNE Template, page 3-7.
- From the Device Types window, as described below.

To add a new device type:

Step 1 Access the VCB, as described in Accessing the VCB, page 3-2.

Step 2 In the VNE Drivers tab, click **Configure Device Types**. The Device Types window appears.

Figure 3-2 Device Types



		Device Type	Category	Device Series	Type
○		Linux	Server		User Defined
○		bbbbbb	Universal Gatew.	Cisco ME 3800X Series Carrier Ethernet Swi	User Defined
○		Cisco 2801	Router		System Default
○		Cisco 7507z	Router		System Default
○		Cisco 3500	Eth-Switch		System Default
○		Cisco Catalyst 3550 12G	Eth-Switch		System Default
○		Cisco ASA 5580	Router		System Default
○		Occam BLC6312	Eth-Switch		System Default
○		Lucent Riverstone 3000	Eth-Switch		System Default
○		Lucent B-STDX8000	ATM-Switch		System Default
○		Cisco Catalyst 2950	Eth-Switch		System Default
○		Cisco 7507mx	Router		System Default
○		Cisco Catalyst 6509 NEB-A	Eth-Switch	Cisco Catalyst 6500 Series Switches	System Default

○ .1.3.6.1.4.1.9.12.3.1.3.720 Cisco Nexus 5020 Switch System Default

The Device Types window provides a table listing all the available device types.

Step 3 Click **Add Row**. A row is added to the device types table, enabling you to define the parameters of the new device type.

Step 4 Define the following parameters:

Parameter	Description
Device Type	The name that will represent the device in the Prime Network GUI, in maps and tables. We recommend that you prefix the device type name with the network element vendor name for easy identification.
Category	The type of device, for example, router, switch, and so on.
Device Series	The device family to which the device type belongs, for example, Cisco 10000 Routers. This will determine which icon will be used to represent the device.
Type	Since you are adding a device type that is not provided as a system default, this field is automatically populated as “User-Defined”.

Step 5 Click **Save**.

Adding Support for Additional Software Versions

Note You cannot edit or delete system default software versions.

The VCB enables you add support for additional software versions and to manage currently supported software versions.

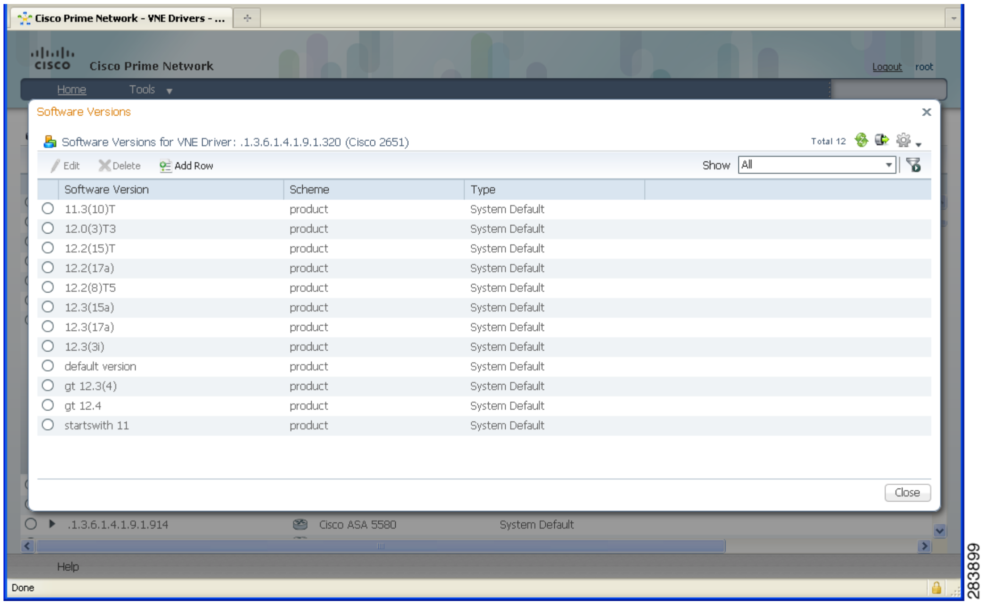
Step 1 Access the VCB, as described in Accessing the VCB, page 3-2.

Step 2 In the VNE Drivers tab, expand the VNE driver for which you want to add a supported software version by clicking on the arrow on the left.

Step 3 Click **Supported s/w versions** in the Software Versions field.

The list of currently supported software for the selected VNE driver is displayed.

Figure 3-3 Software Versions



Software Versions for VNE Driver: .1.3.6.1.4.1.9.1.320 (Cisco 2651)

Software Version	Scheme	Type
○ 11.3(10)T	product	System Default
○ 12.0(3)T3	product	System Default
○ 12.2(15)T	product	System Default
○ 12.2(17a)	product	System Default
○ 12.2(8)T5	product	System Default
○ 12.3(15a)	product	System Default
○ 12.3(17a)	product	System Default
○ 12.3(3i)	product	System Default
○ default version	product	System Default
○ gt 12.3(4)	product	System Default
○ gt 12.4	product	System Default
○ startswith 11	product	System Default

○ .1.3.6.1.4.1.9.1.914 Cisco ASA 5580 System Default

Step 4 Click **Add Row**.

Step 5 Enter the values for the following parameters:

Parameter	Description
New Software Version	The name of the software version to be supported.
Scheme	The scheme based on which the VNE is modeled, either Product or IpCore. For more information about schemes, see the *Cisco Prime Network 4.2.2 Administrator Guide.*
Supported Software Version	The already supported software version to be cloned in order to add support for the new software version.

Step 6 Click **OK**.

Step 7 Test the VNE as described in Testing and Certifying VCB Customizations, page 3-14.

Add Support for Additional Modules

These topics explain how to use the Prime Network VCB to enable support for new standard and pluggable modules on developed VNEs and U-VNEs.

- Adding Support for a New Standard Module, page 3-12
- Adding Support for a New Pluggable Module, page 3-13

If a module does not have "out-of-the-box" support, Prime Network will discover it on a best-effort basis and will generate an informational event with the following description:
"Unsupported module was auto-discovered"

You should validate that all the components of the module were discovered in full. If not, you can add support for the module using the VCB.

Adding Support for a New Standard Module

Module templates are used to enable developed VNEs to recognize new standard modules. Module templates define a set of port layers—from the connector at Layer 0 to encapsulation at Layer 2—that are applicable to a module. Module templates ensure that each port is modeled with the correct port layer information based on the ifType obtained from the SNMP MIB output.

Before You Begin

- Obtain the module identifier and research the capabilities of the module. The module identifier can be found in the properties of the event generated when an unsupported module is auto-discovered.
- Identify the appropriate module template. See Module Templates, page B-65.

To add support for a new standard module:

Step 1 From the Prime Network Administration GUI client, choose **Tools** > **VNE Customization Builder.**

Step 2 Choose **VNE Customization Builder** > **Modules.** In the Modules page, the Cisco Modules tab is selected by default and all the available Cisco module groups are shown. If you are adding support for a non-Cisco module, click the Non-Cisco Modules tab.

Step 3 Click on a module group to show all the modules for that group.

Step 4 Click on the arrow next to a module to expand its display and see all the port layers supported by that module.

If you want to edit a module, select the radio button to the left of the row and then click the **Edit** button If you edit a system default module, the value in the Overriding column will change to **True**. To revert back to the system default values, click **Restore**.

Step 5 Select the module group under which you want to create the new module. All the modules under the module group you choose are displayed.

Step 6 Click the **Add Row** button on the taskbar. A new row opens in an editable mode at the end of the modules display area. Enter the value(s) for the following parameters:

Field	Description
Name	Name of the new module.This field cannot be empty.
Template	Name of the template to be used to enable recognition of the new module. You can select a template from the drop-down list. See Module Templates, page B-65 for information about each template.
Part Number	The part number of the new module.
Hardware Description	The MIB name of the module is usually used as the hardware description. For example, for module .1.3.6.1.4.1.9.12.3.1.9.2.175 (SPA-4XOC48POS/RPR) the hardware description could be cevSpa4pOc48PosSfp. Alternatively, you can use a readable string followed by the mib name, for example, 4 Port OC48 POS SPA (cevSpa4pOc48PosSfp).
Type	The module type can be System Default or User Defined. If you are adding a new module, this field is automatically populated as User Defined and cannot be edited.
Overrides	Indicates if a system defined module has been modified by the user.

Step 7 Click **Save**.

Step 8 Test the new module. See Testing and Certifying Module Customizations, page 3-22.

Adding Support for a New Pluggable Module

This procedure explains how to add support for a pluggable module. When you add support for a new pluggable module, you do not need to specify a template to use.

Note You cannot edit or delete Prime Network default pluggable modules.

To add support for a new pluggable module:

Step 1 From the Prime Network Administration GUI client, choose **Tools** > **VNE Customization Builder.**

Step 2 Choose **VNE Customization Builder** > **Pluggable Modules.** The Pluggable Modules tab displays all the pluggable modules in Prime Network VCB.

If you want to edit a module, select the radio button to the left of the row and then click the **Edit** button

Step 3 Click the **Add Row** button on the taskbar. A new row opens in an editable mode at the end of the pluggable modules display area. Enter the value(s) of the following parameters:

Field	Description
Pluggable Module Name	Name of the new pluggable module.[1] This field cannot be empty.
Media Type	Specifies the media type of the port through which the pluggable module is connected. You can manually enter the values or choose a value from the drop-down list[2]. Only values in drop-down list are valid entries.
PID	The unique Pluggable Module ID. [1]
Pluggable Type	Specifies the type of the new pluggable module. Pluggable types are SFP, XFP, X2, and XENPAK. You can manually enter the values or choose the value from the drop-down list. [2]
Type	Specifies whether the pluggable module is user-defined or Prime Network default. This field by default is “User Defined” and is not editable.

[1] You can enter letters and numbers in pluggable module names and PID. Valid characters for module names are -, ., (,), and +.

[2] When you enter the value in this field, Prime Network VCB GUI displays the values which narrow down as you enter the value.

Step 4 Click **Save**.

Step 5 Test the new module. For the procedure on testing the VNEs, see Testing and Certifying Module Customizations, page 3-22.

Testing and Certifying VCB Customizations

Testing and certifying ensures that you:

- Do not create unintended or undesired results in your production network
- Do obtain the best possible result, a customization that most closely fits technologies, topologies, and other aspects that you need to model

These topics describe the tests you should perform:

- Preparing to Test Your VCB Customizations, page 3-15
- Testing and Certifying U-VNEs, page 3-15
- Testing and Certifying Module Customizations, page 3-22
- Testing and Certifying Event Customizations, page 3-22
- Troubleshooting VCB Customizations, page 3-23

Preparing to Test Your VCB Customizations

Table 3-3 provides a recommended list of test resources for verifying your customizations. Some of these resources are outside of Prime Network.

In addition, the VCB log file is useful for debugging issues that might occur when you use the VCB.

Use the **-logfile** option to specify the name of the log file and the **-debuglevel** option to define the logging level to use. The relevant command is:

```
vcb mode command [command args] -debuglevel INFO -logfile logfilename -user username
-password password
```

When debugging, set the debug level to INFO. The log file is created in the *NETWORKHOME*/Main/logs directory. Error messages are written to the log file and are displayed on the console.

Table 3-3 Testing Tools and Resources

Tool/Resource	Purpose
Prime Network Vision GUI client	Creating a U-VNE instance to verify the physical and logical inventory, that events from the device are received, and ensure the U-VNE does not impact VNE performance.
Direct communication with device	Communicating directly with the device and measuring performance-related issues, such as CPU and memory usage (both with and without the VNE) in various scenarios.
VNE documentation	VCB Template Reference, page B-56, provides the list of technologies and properties that are supported by the template on which the U-VNE is based. *Cisco Prime Network 4.2.2 Supported Cisco VNEs* lists the Prime Network supported devices, software versions, modules, and technologies.
Trap simulator (not supplied with Prime Network).	Enables testing of new events.

Testing and Certifying U-VNEs

After using the VCB to perform device customizations, you must test the VNE or U-VNE to verify that the device and its components can be managed by Prime Network to your satisfaction.To test the results of your VCB commands, we recommend that you add a single instance of the customized VNE to its own AVM in Prime Network.

Examples of the support you should test for include:

- Physical containment
- Logical inventory
- Topology
- Events

We recommend that you record the results of your tests in a compliance report, which certifies the capabilities of the customized VNE. See Preparing Compliance Reports, page 3-18. After you successfully complete the testing process, create additional VNE instances to manage the rest of the devices of this type in the network.

Note When performing the tests described in this section, bear in mind that unlike developed VNEs, U-VNEs are not optimized for a particular device type. The VCB enables you to extend VNEs and create U-VNEs as specified in this document. Other Prime Network features, such as Soft Properties and Command Manager, offer further extensibility to the U-VNE and developed VNE drivers. Prime Network users who need more management capability—or who prefer to have developed VNE drivers for devices that are not already in the Prime Network VNE support scope—can contact their Cisco account representative for any available Cisco Advanced Services alternative.

For details about testing U-VNEs, please see the following sections:

- Setting Up the U-VNE Test Environment, page 3-16
- Performing the U-VNE Tests, page 3-18
- Troubleshooting VCB Customizations, page 3-23

Setting Up the U-VNE Test Environment

Before beginning the testing process, you must ensure that:

- All the mandatory processes—avm11, avm0, and avm100 (only for event support)—are running normally.
- You have access to the device.
- You complete the actions described in Table 3-4.

Table 3-4 Actions to Perform Before You Begin Testing

Action	Purpose	For More Information
Configure the device to send events to the Prime Network server.	This is required before you test whether Prime Network can receive events from the VNE.	*Cisco Prime Network 4.2.2 Administrator Guide*
Create an AVM for hosting the VNE instance on which you will perform the tests.	By isolating the test VNE on its own AVM, you prevent any actions that might impact the functionality of your network.	Creating a Test AVM and VNE, page 3-16
Measure device performance (such as CPU and memory usage) on the device, with and without the VNE.	This enables you to determine whether the VNE is placing an unreasonable load on the device.	Measuring Device Performance, page 3-17
Prepare a compliance report	Fill in the compliance report during testing to track your results against the list of supported technologies and properties outlined in U-VNE Templates, page B-56.	Preparing Compliance Reports, page 3-18

Creating a Test AVM and VNE

We recommend that you create an AVM for the sole purpose of hosting the VNE instance on which you will perform the tests described in this chapter. Placing the VNE instance in its own AVM enables you to test your customizations in a safe environment, where the logs and any errors generated by the VNE are isolated from the rest of your network. This enables you to proceed with testing without having your VNE customizations impact the network, until you are satisfied that the customizations function as required.

Testing a single VNE instance also helps you scale a rollout more easily. For example, if you have 100 devices of a certain type that you are introducing to your managed network, first create a single VNE instance on which to test your customizations. After testing is complete, create additional VNEs for the other 99 devices.

AVMs and VNEs can be created in the Prime Network Administration client. For further details about creating an AVM and adding a VNE, see the *Cisco Prime Network 4.2.2 Administrator Guide*.

Checking the Communication and Investigation State of the Test Instance

After you create a test instance of the U-VNE, you must verify that Prime Network can communicate with the device (communication state) and that Prime Network can discover the physical and logical inventory of the device (investigation state).

To check the communication and investigation state of the test instance:

Step 1 Log into Prime Network Vision.

Step 2 Choose **Network Inventory > Network Elements**.

Step 3 Perform a search to locate the device.

Step 4 Check the Communication State column to see the status of the communication between Prime Network and the device:

- If the status is Device Reachable, this indicates that all of the enabled protocols on the device are responding.
- If the status is Device Unreachable, this indicates that at least one of the enabled protocols is not responding. If this occurs, troubleshoot the problem, as described in Device Unreachable, page 3-23.

Step 5 Check the Investigation State column to see how successfully the VNE has modeled the device it represents:

- If the status is Operational, proceed to the next test.
- If the status is Incomplete, this indicates that Prime Network is unable to model all the components in the device. You must investigate further to determine whether the components or properties that cannot be modeled prevent you from using the U-VNE in your network. For troubleshooting details, see Investigation State Not Operational, page 3-23.

Note For more information on troubleshooting VNE modeling, see the *Cisco Prime Network 4.2.2 Administrator Guide*.

Measuring Device Performance

We recommend that you use an element manager or similar application to measure CPU and memory usage on the device before you start managing it with the VNE. This provides a baseline for comparison when you later measure the load on the VNE as part of the testing process. You should simulate various scenarios, including those that place heavy loads on the device, so that you can later determine the effect of your VNE customizations on device performance. See Testing CPU and Memory Usage, page 3-21.

Performing the U-VNE Tests

To test and certify U-VNEs, we recommend that you follow these procedures.

1. Preparing Compliance Reports, page 3-18
2. Testing the Physical Inventory, page 3-18
3. Testing the Logical Inventory, page 3-20
4. Testing CPU and Memory Usage, page 3-21
5. Verifying That Prime Network Receives Events from an NE and Generates Tickets, page 3-21

Preparing Compliance Reports

As you complete the tests described in this chapter, we recommend that you capture the results in a compliance report. A compliance report makes it easier for you to evaluate whether the U-VNE suits your needs or whether you will need to try again by cloning from a different device or using a different U-VNE template.

Create a compliance report using the information provided in *Cisco Prime Network 4.2.2 Supported Cisco VNEs* (for a U-VNE that was cloned from an existing device family) or U-VNE Templates, page B-56 (for a template-based U-VNE). Use the information to do the following:

1. Copy each technology table to a file (such as a spreadsheet).
2. Remove any columns that describe support for templates other than the one you are using.
3. Add a column for verifying whether each supported property is modeled in Prime Network.
4. Fill out the final column, identifying the supported and unsupported properties.

Assessing a customization requires that you test on a reasonable configuration (test-to-success) and that you evaluate extreme cases (test-to-fail).

Note You cannot use the VCB to change the templates themselves; for example, to support additional CLIs or MIBs or to modify the parsing of the device results.

If you find gaps in the modeling of the U-VNE, you can try using Soft Properties. If you need more management capability or you prefer to have developed VNE drivers for devices that are not already in the Prime Network VNE support scope, contact your Cisco account representative for any available Cisco Advanced Services alternative.

Testing the Physical Inventory

The physical inventory reflects the physical components of the managed device, including its:

- Port Connector—Port details.
- Shelf—Chassis or rack where various types of equipment may be placed or inserted.
- Slot—Details on where the modules are inserted.
- Module—Physical module or adapter card details; hardware description.

Note The list of components that are actually displayed for the U-VNE in the Prime Network client are dependent on the device from which the U-VNE was cloned or on the template that was used to create the U-VNE.

To view the physical inventory:

Step 1 Log into Prime Network Vision.

Step 2 Choose **Network Inventory > Network Elements**.

Step 3 Perform a search to locate the device.

Step 4 Double-click the device.

Step 5 Expand the **Physical Inventory** tree node.

Step 6 Compare the physical containment displayed in the Physical Inventory tree with the actual components contained in the device managed by the U-VNE. In particular, make sure that the following information is accurate:

- Slot Number
- Number of Ports
- Connector Type

Step 7 Look for signs that the U-VNE does not fully model the device, including:

- Investigation state is Currently Unsynchronized.
- A module without ports; for example, if the WS-X6724 module is described as having 24 ports, but no ports appear beneath the module in the tree.

Step 8 Select each component in the tree, then check the Properties window to verify that the physical inventory includes all the properties that are supported by the U-VNE (as defined in the U-VNE template on which it is based).

Step 9 Record your results in a compliance report. See Preparing Compliance Reports, page 3-18.

Step 10 If there are components or properties that are missing or incomplete, you must decide whether these components or properties prevent you from using the U-VNE in your network:

- If you determine that the U-VNE can be managed successfully by Prime Network without the unsupported component or property, proceed to the next test.

Tip In the case of a missing property, try adding it using the Soft Property Builder.

- If you need more management capability or you prefer to have developed VNE drivers for devices that are not already in the Prime Network VNE support scope, you can contact your Cisco account representative for any available Cisco Advanced Services alternative.

  (The VCB enables you to extend VNEs and create U-VNEs as specified in this chapter. Other Prime Network features, such as Soft Properties and Command Builder, described in separate chapters, offer further extensibility to the U-VNE and developed VNE drivers.)

Testing the Logical Inventory

The logical inventory reflects dynamic data such as configuration data, forwarding, and service-related components that affect traffic handling in the element. The list of components that are actually displayed in the logical inventory are dependent on the technologies supported either the device family from which the U-VNE was cloned or by the template that was used to create the U-VNE.

To view the logical inventory:

Step 1 Log into Prime Network Vision.

Step 2 Choose **Network Inventory > Network Elements**.

Step 3 Perform a search to locate the device.

Step 4 Double-click the device.

Step 5 Expand the **Logical Inventory** tree node.

Step 6 Compare the information displayed in the Logical Inventory tree with the actual technologies supported by the U-VNE, as defined by the template on which the U-VNE is based. See U-VNE Templates, page B-56.

Step 7 Select each component in the tree, then check the Properties view to verify that the logical inventory includes all the properties that are supported by the U-VNE (as defined in the U-VNE template on which it is based).

Verify the IP interfaces by querying the device for its list of IP interfaces and verifying that all of them appear under IP Flow Points. Verify that all the IP interfaces configured on the device appear in the IP Interfaces tab under Routing Entity.

Step 8 Record your results in a compliance report. See Preparing Compliance Reports, page 3-18.

Step 9 If any technologies or properties are missing or incomplete, you must investigate further to determine whether these technologies or properties prevent you from using the U-VNE in your network:

- If you determine that the device can be managed successfully by Prime Network without the unsupported technology or property, proceed to the next test.
- If you need more management capability or you prefer to have developed VNE drivers for devices that are not already in the Prime Network VNE support scope, you can contact your Cisco account representative for any available Cisco Advanced Services alternative.

Testing CPU and Memory Usage

Note Ensure that you perform both positive testing (on a reasonable configuration) and negative testing (on simulations of expected network scenarios).

In addition to verifying how well the U-VNE models the device, we recommend that you measure the CPU and memory usage demands placed by the U-VNE on the device. During test preparation, before adding the VNE instance, you measured CPU and memory usage on the NE; (see Measuring Device Performance, page 3-17). Now, compare the usage on the NE against the usage for the VNE instance on the Prime Network AVM and unit as follows:

- To view CPU usage, look at the properties of the device in the Inventory window.
- To monitor additional information, such as memory usage, use the Prime Network diagnostic client. To access the diagnostic client, enter the following address in your web browser (the username is normally admin; you will have to get the password from your system administrator):

  https://*gateway-IP-address*:1311

Verifying That Prime Network Receives Events from an NE and Generates Tickets

This topic provides steps for verifying that Prime Network receives events from an NE and that the VNE driver parses the events correctly, generating tickets if the events are ticketable.

Note Event parsing depends on the completeness and correctness of the modeling. Parsing BGP events, for example, depends on BGP being modeled correctly in the inventory.

Before You Begin

Make sure that you have configured the device to send events to the Prime Network server and created a Link Down event.

To check for events from the NE:

Step 1 Log into Prime Network Vision.

Step 2 Add the device to a map.

Step 3 Right-click on the device in the map and select Filter Tickets. The tickets pane below the map will show the tickets for the selected device.

Step 4 Check whether the Link Down event that you generated during your test preparations appears in the table.

Step 5 Generate additional events from the device, then see if they appear in the events table.

Step 6 If the events do not appear in the events table, proceed as follows:

- Check for mistakes in the device configuration.
- Use an external tool, such as a MIB browser, to determine whether events are being sent by the device.
- Troubleshoot event customization, as described in Customizing Events, page 6-15.

Testing and Certifying Module Customizations

First perform a static test of the module, using the **vcb view module** command. If the module is not correctly configured, delete it and add the module again.

After verifying the module statically, set up your test environment just as you would for testing device customization—see Setting Up the U-VNE Test Environment, page 3-16. Then view the physical inventory for your U-VNE test instance as follows.

Step 1 Launch Prime Network Vision.

Step 2 Add the containing device to a map.

Step 3 Double-click the device.

Step 4 Expand the **Physical Inventory** tree node.

Step 5 Confirm the correctness of the following for the module:

- Slot Number
- Number of Ports
- Connector Type

Step 6 Move customizations into production during a maintenance window. See Testing and Certifying U-VNEs, page 3-15.

Testing and Certifying Event Customizations

It is recommended that you test and certify event customizations in your lab before moving them to production:

To receive events from a device, you must configure the device with the details of the Prime Network server. For example, use the following commands for devices running Cisco IOS or Catalyst OS software:

```
snmp-server host 172.20.2.160
logging trap informational
logging source-interface Loopback0
logging on
logging 172.20.2.160
```

A similar set of commands should be used for devices belonging to other manufacturers.

These commands enable the device to send traps to the specified gateway IP over port 162. Therefore, port 162 must be enabled to receive traps from the device. In addition, you must reserve port 1162 for the general trap processing by the Prime Network server. This task is handled by the AVM 100 process.

To test your event:

Step 1 Send an event from an NE or from a simulator. Open Prime Network Events and verify that:

- – Prime Network recognizes and processes the event
- – Event parameters—type, subtype, severity, and so on—are as expected
- – Unique ID is appended to source ID (ManagedElement)

After you complete the tests, move the customization to production and test there as well; see Testing and Certifying Event Customizations, page 3-22.

Troubleshooting VCB Customizations

This section describes basic troubleshooting procedures to perform when Prime Network cannot communicate with a U-VNE, and includes the following topics:

- Device Unreachable, page 3-23
- Investigation State Not Operational, page 3-23

Device Unreachable

If you cannot communicate with a U-VNE, try the following:

- Launch Prime Network Administration and verify the Admin Status and Operational Status of the VNE. The Admin Status should be Enabled and the Operational Status should be Up.
- Verify that you are using the correct SNMP and Telnet credentials for the device.
- Ping the device from the Prime Network server.

Note For more information, see the *Cisco Prime Network 4.2.2 Administrator Guide*.

Investigation State Not Operational

If the investigation state of the U-VNE is any value other than Operational, perform the actions described in the following table.

Table 3-5 Troubleshooting the Investigation State

State	Description	Action
Partially Synchronized	The U-VNE is encountering a problem, such as an exception caused by a particular discovery command or an unsupported module.	Examine the AVM log for messages about: • Failed commands or OIDs. • Unsupported modules. See Log Entry for Unsupported Module, page 3-24. **Note** For more information, see Prime Network Logs in the *Cisco Prime Network 4.2.2 Administrator Guide*. Check Prime Network Vision or Events for an “unsupported module” system event.
Unsupported	The U-VNE is encountering registration problems.	Use the **vcb sitechanges view** command to verify that the U-VNE registrations were created properly. In particular, make sure that you used the correct sysObjectID for this device type when using the VCB to create the U-VNE-driver. If the sysObjectID is not correct: • Delete the U-VNE. • Recreate the U-VNE driver using the correct sysObjectID. See Creating a U-VNE Using the GenericUVNE Template, page 3-7.

Log Entry for Unsupported Module

The log entry for an unsupported module looks like this:

```
ERROR [07 07 2010 17:07:47.810 IST] - PhysicalCommandHandler.isEntitySupported -
192.168.20.1 can't create module with entry
value spec/physical/modules/.1.3.6.1.4.1.9.12.3.1.9.29.118.1/loaders will use
default module loader
```

If you have an unsupported module, your choices are to either use the VCB to add the module (see Adding Support for a New Pluggable Module, page 3-13), or manage the device in Prime Network without managing that particular module.

Deploying VCB Customizations to Your Production Environment

Tip Always perform customization in a maintenance window.

During a maintenance window, recreate the extensions that you have tested and certified in your lab by following this procedure:

	Action	For more information
Step 1	(Optional) View the extensions that exist in your lab system, taking note of those that you want to deploy in your Prime Network production system.	Viewing Existing VCB Customizations in the Registry, page 3-25
Step 2	Export the extensions from the Prime Network gateway in the lab to the VcbImportCommands.sh file, located in *NETWORKHOME*/Main.	Exporting VCB Customizations to Another Gateway (GUI), page 3-27
Step 3	Import the VcbImportCommands.sh file to the Prime Network gateway in the production environment. If necessary, you can remove any extensions that you do not want to deploy from the script.	Importing VCB Customizations from Another Gateway (GUI), page 3-27
Step 4	Restart all AVMs and VNEs that support the customizations.	
Step 5	Repeat testing and certification to ensure that the customizations are functioning as expected in your production environment.	Performing the U-VNE Tests, page 3-18

Viewing Existing VCB Customizations in the Registry

To view registry changes to the site.xml file made with the VCB, enter the following command:

```
vcb sitechanges view -user username -password password
```

Here is an example of the **vcb sitechanges view** command output; this examples shows customizations that were made, including some event patterns, event parsing rules, and events.

```
# [~/Main/registry/ConfigurationFiles]% vcb sitechanges view -user username -password
password


>>>>>>>>>>>>>>>>>>>>>Event Pattern<<<<<<<<<<<<<<<<<<<<<<<<<

Hive Name:cisco-syslog-ipcore-parsing-rules
```

```
Group: cisco-syslog-ipcore-parsing-rules
Pattern ID: 5004
    Rule Name: FWSM-5-713131
    Repository: cisco-syslog-repository
    User-defined: true
Pattern ID: 5005
    Rule Name: FWSM-4-109022
    Repository: cisco-syslog-repository
    User-defined: true

Hive Name:cisco-trap-product-parsing-rules
Group: cisco-trap-product-parsing-rules
Pattern ID: 5008
    Rule Name: STACKWISE-MEMBER-STATUS
    Repository: cisco-trap-repository
    User-defined: true
>>>>>>>>>>>>>>>>>>>>>Event Parsing Rules<<<<<<<<<<<<<<<<<<<<<<<<<
Hive Name:cisco-trap-repository
--------------------------------------------------------------------
RuleName                          Description
--------------------------------------------------------------------
STACKWISE-MEMBER-STATUS           STACKWISE-MEMBER-STATUS
--------------------------------------------------------------------
total rows in report: 1

Hive Name:cisco-syslog-repository
--------------------------------------------------------------
RuleName                       Description
--------------------------------------------------------------
FWSM-4-109022                  FWSM-4-109022
FWSM-5-713131                  FWSM-5-713131
--------------------------------------------------------------
total rows in report: 2

>>>>>>>>>>>>>>>>>>>>>UVne<<<<<<<<<<<<<<<<<<<<<<<<<

>>>>>>>>>>>>>>>>>>>>>Event<<<<<<<<<<<<<<<<<<<<<<<<<
*****
Event Name: stackwise status trap
    Alarm ID: 9061
    User Defined: true
    Subtype: stack member removed
        Severity: WARNING
        Short Description: Stack Member Removed
        Ticketable: true
        Auto Clear: true
    Subtype: stack member added
        Severity: CLEARED
        Short Description: Stack New Member
        Ticketable: false
        Auto Clear: false
*****
Event Name: DWDM fatal error 2 syslog
    Alarm ID: 1341
    User Defined: false
    Subtype: DWDM fatal error 2 syslog
        Severity: MAJOR
        Short Description: DWDM fatal error with reason and error number
        Ticketable: true
        Auto Clear: false
```

Exporting VCB Customizations to Another Gateway (GUI)

You can export your VCB customizations and then import them to another Prime Network gateway. During export, all customizations in the registry will be exported, including all user-defined items and all system default overrides.

To export VCB customizations:

Step 1 In the VCB tool, click **Export Customization**. The customizations are exported from site.xml to a VCB command script file, VcbImportCommands.sh, located in *NETWORKHOME*/Main.

Step 2 Save the file.

Importing VCB Customizations from Another Gateway (GUI)

To import VCB customizations:

Step 1 In the VCB tool, click **Import Customization**.

Step 2 Select the VcbImportCommands.sh file to be imported. The full script is displayed in the Import Customizations dialog.

Step 3 Optional. Review the script and remove commands that you do not want to import. Click **Next**.

Step 4 Enter the required credentials, as follows:

Field	Description
Gateway Username and Password	The username and password used to access Prime Network components, including the gateway.
Prime Network Username and Password	The username and password used when logging into the Prime Network GUI.

Step 5 Click **Next**. The import process begins. When it is complete, you will receive confirmation and any errors that occur during the execution will be displayed.

Step 6 Check that the imported customizations appear in the VCB tool.

Importing VCB Registry Customizations (CLI)

To import VCB registry customizations:

Step 1 After you export registry changes, copy the VcbImportCommands.sh file to the *NETWORKHOME/Main* folder on the Prime Network gateway on which you want to import the customizations.

Step 2 (Optional) Edit the VcbImportCommands.sh file and delete any customizations that you do not want to import. The commands of interest in the file start with **$VCBPATH** as shown in this example:

$VCBPATH eventpattern add -rulename L2-DWDM-3-FATAL_2
-group cisco-syslog-ipcore-parsing-rules -repository cisco-router-iox-syslog-repository -user $USER
-password $PASS >> **"$VCT_IMPORT_SCRIPT_LOG"**

Step 3 Change permissions on the script to ensure that it is executable, by entering a command such as this one:

chmod 755 VcbImportCommands.sh

Step 4 Run the script from the *NETWORKHOME/Main* folder, by entering this command:

VcbImportCommands.sh *username password*

Deleting VNE Customizations from the Registry

These topics describe how to roll back a registry configuration to its original settings, and how to completely delete customizations from the registry.

Customizations using the VCB affect VNE drivers and update the Prime Network registry in a safe manner. The VCB enables you to roll back easily; you can remove:

- All VCB customizations with one command, restoring your system to a factory-defined state.
- Selective VCB customizations, using one command per customization that you want to remove.

Because VCB customizations are carried forward during an upgrade to a new version of Prime Network, your customizations continue to override any new or updated VNE drivers or newly supported events and modules. The ability to remove changes selectively enables you to discontinue particular overrides only and take advantage of any newly added support.

To delete VNE customizations:

Step 1 Create a script file,VcbDeleteCommands.sh, in *NETWORKHOME*/Main by entering this command:

```
vcb sitechanges delete -user username -password password
```

Step 2 (Optional) To retain any specific customizations, edit the VcbDeleteCommands.sh file and remove any line that deletes a customization that you want to keep. The commands of interest in the file start with **$VCBPATH** as shown in this example:

```
$VCBPATH eventpattern delete -group cisco-syslog-ipcore-parsing-rules -patternid 5001
-user $USER -password $PASS >> "$VCT_IMPORT_SCRIPT_LOG"
```

Step 3 Change permissions on the script to ensure that it is executable, by entering a command such as this one:

```
chmod 755 VcbDeleteCommands.sh
```

Step 4 Run the script from the *NETWORKHOME*/Main folder, by entering this command:

```
VcbDeleteCommands.sh user username password password
```

PART 3

Extending Device Configuration Capabilities

These topics describe how to perform customized device configuration operations:

- Creating Commands and Command Scripts to Perform Device Configuration Operations, page 4-1
- Deploying Activation Workflows to Devices Using Transaction Manager, page 5-1

Creating Commands and Command Scripts to Perform Device Configuration Operations

Prime Network provides two tools for creating commands and scripts that perform device configuration operations: Command Builder and Command Manager. Commands can range from simple show commands to command scripts that contains wizards with multiple pages and input methods, such as check boxes and drop-down lists. After you create command scripts, you can add them to the Vision GUI client. Users with the required privileges can run the command scripts by right-clicking an NE's **Commands** menu. To get started, see these topics:

- How is Command Manager Different From Command Builder?, page 4-1
- Checking Global Command Settings Before You Begin Using Command Manager, page 4-2
- Using Command Manager to Create and Execute Device Configuration Command Scripts, page 4-3
- Using Command Builder to Create Device Configuration Command Scripts, page 4-21

How is Command Manager Different From Command Builder?

Command Manager uses the Command Builder back-end, but unlike Command Builder, it also provides a command repository that contains all of the device-level commands and command scripts on the Prime Network gateway. Command Manager also allows you to execute commands on multiple devices at one time. You can organize commands by assigning versions to commands, or group commands into user-specified packages. For bulk configurations, you can do parameter substitution by importing data from a file.

Command Manager supports creating commands and command scripts at the network element level (IManagedElement). Therefore, the command repository only displays scripts that are available at the device level. To create commands at other levels, use Command Builder.

Checking Global Command Settings Before You Begin Using Command Manager

While Command Manager and Command Builder let you specify the required user access level for commands, Prime Network has some global settings which will also affect authentication and authorization:

- Whether users must enter their credentials before running command scripts
- Whether users can schedule device configuration jobs
- Whether a warning message is displayed when users run command scripts

These settings are controlled from the Administration GUI client and are described in the following sections.

Controlling Who Can Run Command Scripts

Users can execute commands from the Vision GUI client when a device is in their scope and they have sufficient privileges. You specify the required privileges when you create the command's basic information. However, the following two settings in the Administration GUI client will also affect command operation. (These are in the Administration GUI client under **Global Settings > Security Settings > User Account Settings**.)

Global Setting	How It Affects Command Manager and Command Builder
Execution of Commands	If enabled ("Ask for user credentials when running device configuration operation" check box is checked), requires users to enter their credentials when they execute a command from a device's right-click Commands menu in the Vision GUI client, or when commands are run from Command Manager. This only applies to commands that are executed *immediately* (it does not apply to scheduled commands). Provisioning and Audit events display an additional column that lists the username. This setting is disabled by default.
Job Scheduling	Enables and disables the global per-user job authorization mode. This affects Command Manager jobs, and commands that are *scheduled* using the device right-click Commands menu in the Vision GUI client. If the mode is enabled, job scheduling privileges are controlled by the setting in the individual user accounts. • If this mode is enabled and a user is granted privileges, the user can schedule jobs across the product. • If this mode is enabled and a user is not granted privileges, the job scheduling features in the user's GUI clients are disabled. If the global per-user authorization mode is disabled, all users can schedule jobs; the setting in the users's account is ignored. This mode is disabled by default (all users can schedule jobs).

For more information on these global settings, see the *Cisco Prime Network 4.2.2 Administrator Guide*.

Displaying a Warning Message When Users Run Commands

Prime Network can be configured to display a warning message whenever users execute commands. This affects Command Manager jobs, and any commands that are executed using the device right-click Commands menu in the Vision GUI client.

Users must acknowledge the message before proceeding. This feature is disabled by default. To enable it and specify a message, choose **Global Settings > Commands** in the Administration GUI client, enable the feature, and enter the message text. When users run commands, they will have to acknowledge the warning message or Prime Network will not allow them to proceed.

Using Command Manager to Create and Execute Device Configuration Command Scripts

These topics describe how to use Command Manager to create device configuration commands and command scripts:

- What is Command Manager?, page 4-3
- Steps for Creating and Deploying Commands with Command Manager, page 4-4
- Command Manager Window and Repository, page 4-5
- Creating and Testing a Command or Command Sequence, page 4-7
- Running Commands Using Input Stored in a File, page 4-18
- Controlling Command Manager Jobs, page 4-19
- Moving, Importing, and Exporting Commands and Packages, page 4-19
- Deleting Commands and Packages, page 4-20
- Deleting Commands and Packages from the Prime Network Command Repository, page 4-21

What is Command Manager?

Like Command Builder, Command Manager provides a GUI framework for creating programmable SNMP or Telnet commands. Commands can include multiple input screens and multiple input methods—text fields, check boxes, drop-down lists, and so forth. Commands can include parameters that are fixed. You can also perform bulk configurations using XLS or CSV files to populate the parameter values

Command Manager uses the Command Builder back-end, but it provides the following additional features:

- View the command repository, which contains all device-level commands available on the system
- Execute commands on multiple devices
- Use input from files to provide parameter substitution
- Apply versions to individual commands
- Create command packages

However, Command Manager commands can only be applied at the network element (IManagedElement) level, while Command Builder commands can be associated with any existing object group (IMO), type, or instance. This is why only device-level commands are listed in the command repository.

Once a command is created, it is added to the central repository and can be made available to other NEs of the same family, NE type, or software version, and can be executed from the Vision GUI client by users according to their device scope and privileges. Users with Configurator privileges can also create simple command sequences, although command sequences can only be run from the command repository (not from the Vision GUI client).

Command information is stored in XML files that can be organized into packages. The Command Manager repository always contains these two packages:

- **Default**—Contains all commands that are provided with the Prime Network product. Out-of-the-box commands in the Default package cannot be moved or exported.
- **Production**—Contains any user-created commands that were migrated from another installation. For a new installation, the Production package will be empty.

As you create new commands, you can place them in new packages and reorganize them as needed. You can also export and import command packages. Commands installed from Device Packages are listed only if Prime Network is restarted.

Command Manager jobs are purged according to the Prime Network job purging settings. Commands information (including rules, parameters, and script implementations) are stored on the gateway and are not purged; only Command Manager *jobs* are purged.

Commands created with Command Manager can be used by Transaction Manager. Commands can also be integrated into external configuration applications using the Prime Network API. For information on how to do this, see the *Cisco Prime Network Integration Developer Guide*.

Steps for Creating and Deploying Commands with Command Manager

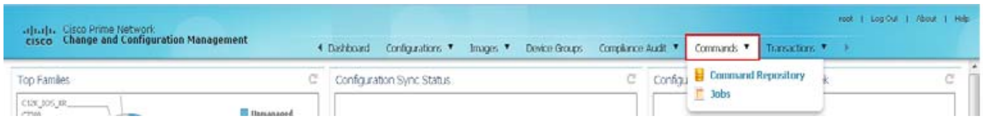
The following table lists the steps you must follow to create a command or command sequence.

	Step Description	**See:**
Step 1	Check the settings that may affect who can run commands you create, and whether a warning message is displayed when users run a command.	Checking Global Command Settings Before You Begin Using Command Manager, page 4-2
Step 2	Create a package for the command, giving it a name that is meaningful for its contents (such as a device type). We recommend not using the Default package to keep user-created commands separate from out-of-the-box commands (for ease of management).	Creating a New Package, page 4-7
Step 3	Provide a name for the command (that will appear in the NE right-click menu) and specify the required user access level (this is the command's basic information).	Defining a Command's Basic Information, page 4-8

	Step Description	See:
Step 4	Define the properties that determine the format and structure of the command input form. These are called the *Execution Parameters*. For each parameter, define the type of input the user must provide (a string, a choice from a drop-down list, a checked check box, and so forth). A command can have multiple parameters, or multiple input pages (tabs), each with their own parameters. If you use Bean Shell scripts in the next step, you can use the script logic to populate and validate the parameter values.	Defining Execution Parameters for a Command, page 4-9
Step 5	Specify the devices and software versions that the command can be run on, along with the scripts that should run when a user executes the command. These are called *Command Implementations*. Scripts can use: • Bean Shell over Telnet, SNMP, or TL1; or • Cisco Prime Network Macro language (over Telnet only). You can also specify rollback scripts and the conditions that indicate the command has failed.	Defining the Implementations (Scripts) for a Command, page 4-12
Step 6	Test the command locally by previewing each implementation.	
Step 7	Create the command.	
Step 8	Test the command to see how it will appear to the user, review the script execution, and execute the command (which deploys the command to applicable devices).	Previewing a Command Script and Checking Command Job Results, page 4-16
Step 9	Check the command job status.	Controlling Command Manager Jobs, page 4-19

Command Manager Window and Repository

Launch the Command Manager from Change and Configuration Management, as shown in the following figure. You may have to expand the title bar by clicking the arrow at the top left.



361868

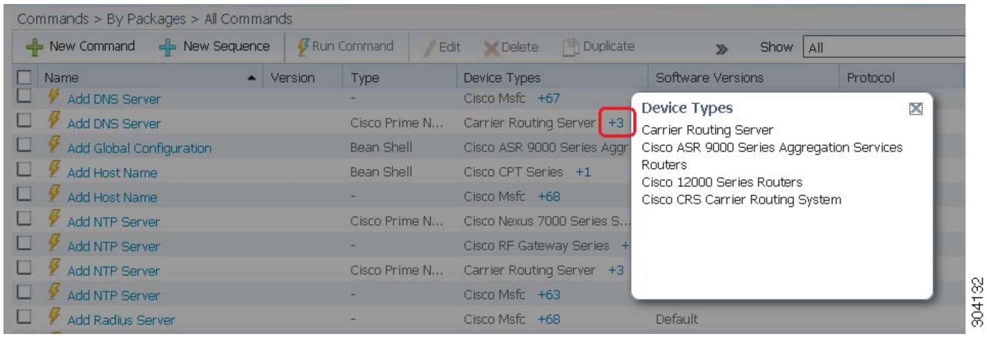
When you select **Command > Command Repository**, Prime Network displays all of the device-level commands in the system. The Default package contains the out-of-the-box commands provided with Prime Network. The Production package is only populated in upgrade scenarios and contains the commands that were migrated from the older deployment. Commands can be executed from the repository or from the devices (in the Vision GUI client). Commands displayed in the repository are filtered according to the user's access level.

The command repository only lists device-level commands (that is, commands at the IManagedElement level). For commands at other levels, use Command Builder.

Commands are indicated by a yellow lightning bolt and sequences by a blue lightning bolt. You can get the following information from the repository:

- Find out whether the command can be executed on different device types. This is indicated by a hyperlink in the Device Type column, as shown in the following figure. In this example, the Add DNS Server command can be executed on all CRS, ASR 9000, and 12000 routers. This also indicates that if those devices appear in a Vision GUI client map, the Add DNS Server command can be executed from the device's right-click Commands menu.

Figure 4-1 Command Manager Repository Showing Applicable Device Types for a Command

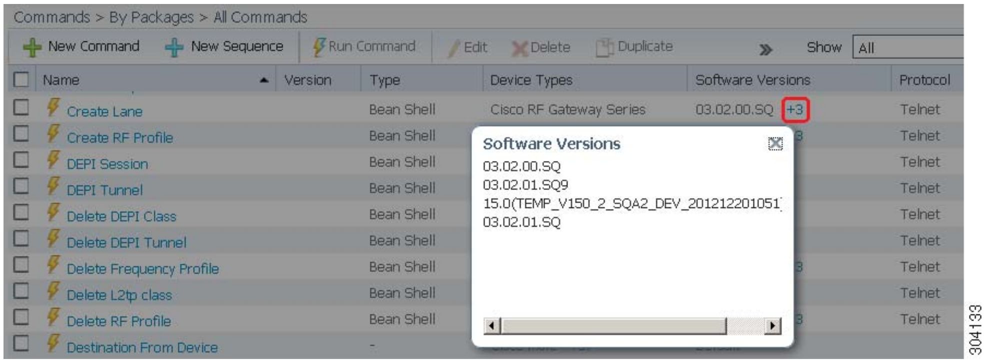


□	Name	Version	Type	Device Types	Software Versions	Protocol
□	Add DNS Server		-	Cisco Msfc +67		
□	Add DNS Server		Cisco Prime N...	Carrier Routing Server +3		
□	Add Global Configuration		Bean Shell	Cisco ASR 9000 Series Aggr		
□	Add Host Name		Bean Shell	Cisco CPT Series +1		
□	Add Host Name		-	Cisco Msfc +68		
□	Add NTP Server		Cisco Prime N...	Cisco Nexus 7000 Series S...		
□	Add NTP Server		-	Cisco RF Gateway Series +		
□	Add NTP Server		Cisco Prime N...	Carrier Routing Server +3		
□	Add NTP Server		-	Cisco Msfc +63		
□	Add Radius Server		-	Cisco Msfc +68	Default	

Device Types
Carrier Routing Server
Cisco ASR 9000 Series Aggregation Services Routers
Cisco 12000 Series Routers
Cisco CRS Carrier Routing System

- Find out if the command can be executed on multiple software versions (for a single device type). This is indicated by a hyperlink in the Software Version column. In the following example, the Create Lane command can be executed on a Cisco RF Gateway Series device that is running any of the software versions that are listed.

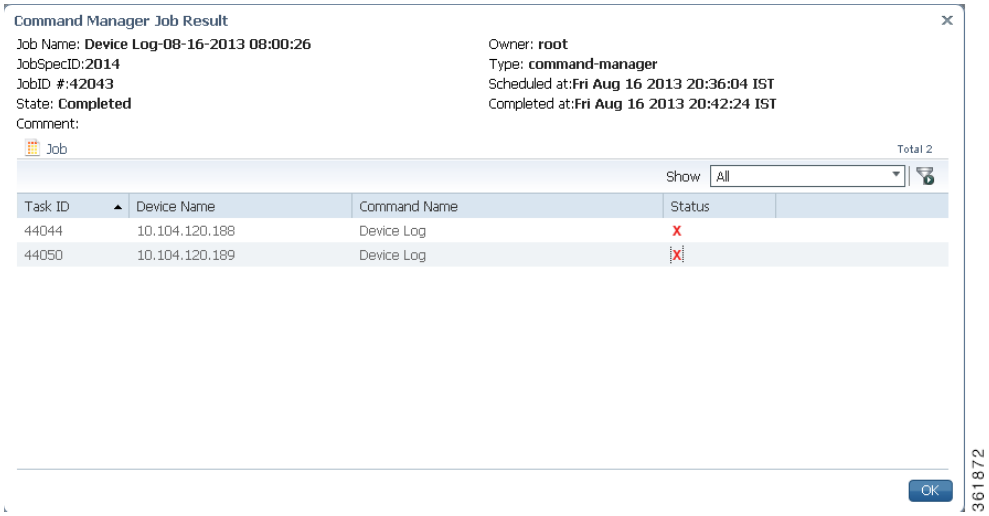
Figure 4-2 Command Manager Window Showing Applicable Software Versions for a Command



□	Name	Version	Type	Device Types	Software Versions	Protocol
□	Create Lane		Bean Shell	Cisco RF Gateway Series	03.02.00.SQ +3	Telnet
□	Create RF Profile		Bean Shell		3	Telnet
□	DEPI Session		Bean Shell			Telnet
□	DEPI Tunnel		Bean Shell			Telnet
□	Delete DEPI Class		Bean Shell			Telnet
□	Delete DEPI Tunnel		Bean Shell			Telnet
□	Delete Frequency Profile		Bean Shell		3	Telnet
□	Delete L2tp class		Bean Shell			Telnet
□	Delete RF Profile		Bean Shell		3	Telnet
□	Destination From Device		-			

Software Versions
03.02.00.SQ
03.02.01.SQ9
15.0(TEMP_V150_2_SQA2_DEV_201212201051
03.02.01.SQ

- Find out the number of jobs associated with the command. This is indicated by a hyperlink in the # Of Jobs column which displays the Command Manager Jobs page. Depending on the user access role the jobs are displayed for the selected command.
- Find out the last run result of the command. This is indicated by a hyperlink in the Last Run column, as shown in the following figure. In the following example, the details of each job and its result associated with the command Device Log is displayed.



Job Name: Device Log-08-16-2013 08:00:26
JobSpecID:2014
JobID #:42043
State: Completed
Comment:

Owner: root
Type: command-manager
Scheduled at:Fri Aug 16 2013 20:36:04 IST
Completed at:Fri Aug 16 2013 20:42:24 IST

Task ID	Device Name	Command Name	Status
44044	10.104.120.188	Device Log	X
44050	10.104.120.189	Device Log	X

Users with Administrator and Configurator privileges can perform all operations on all commands—editing, executing, deleting, moving, and so forth. All other users can perform these operations if they have the required privileges for the command, and the device is in their scope.

Creating and Testing a Command or Command Sequence

Commands and sequences must be created within a package. A best practice is to create a package with a name that describes the contents, such as the device series the command can be run on.

Commands can be duplicated and edited, but only if they are user-created commands. You cannot duplicate out-of-the-box commands that are packaged with Prime Network.

When you create a command (by clicking **Create** in the Command Implementations area), the command is saved to the database. It is not deployed to any NEs until you go to the repository, select the command, and click **Run Command**.

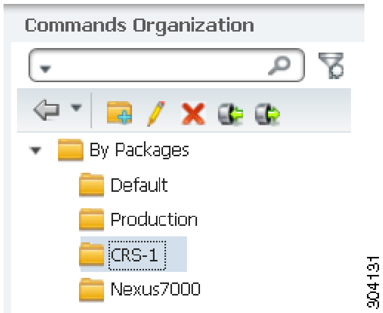
These topics provide the steps required to create, test, and deploy commands and command sequences:

- Creating a New Package, page 4-7
- Creating a New Command, page 4-8
- Creating a Command Sequence, page 4-15
- Previewing a Command Script and Checking Command Job Results, page 4-16

Creating a New Package

Use the package mechanism to organize your command scripts. Naming packages by device type is a good practice, as shown in the following figure. If you need to create a new package, click the New Package icon on the upper left side of the window and provide a name and optional description.

Figure 4-3 Command Packages



If you delete any of the user-created packages (such as CRS-1 or Nexus7000 in the previous figure), all commands associated with package are moved to the Default package. This ensures that there will be no problems if the command is part of a scheduled job.

New commands that are installed from a Device Package are not listed in the command repository unless Prime Network is restarted. (Remember that the command repository only lists device-level commands; that is, commands at the IManagedElement level.)

To create a new command, proceed to the next section. To create a command sequence, see Creating a Command Sequence, page 4-15.

Creating a New Command

These are the steps for creating a new command script with Command Manager:

- Defining a Command's Basic Information, page 4-8
- Defining Execution Parameters for a Command, page 4-9
- Defining the Implementations (Scripts) for a Command, page 4-12

Defining a Command's Basic Information

The basic information for a command includes the name that will appear in the Vision GUI client (in the NE right-click menu) the user authorization that is required to use the command.

Step 1 Select a package from the Commands Organization area.

Tip Do not use the Default package because it contains all commands provided by Prime Network. This makes it easier to your manage user-created commands.

Step 2 Click **New Command** to open the command wizard builder.

Step 3 Configure the command's basic information in the Command Info area.

Command Info Field	Description
Command Title	The name of the command. This is what will be displayed in the Vision GUI client when a user right-clicks an NE and chooses **Commands**.
Menu Category	The menu under which the command should be listed. This specifies where the command is displayed when a user right-clicks the NE in the Vision GUI client. For example, for a command named *new-command*: • **Commands**—Would render **Commands** > *new-command* • **Commands/Custom**—Would render **Commands** > **Custom** > *new-command*
Version	A number or letter which helps you track instances of the same command. A best practice is to change this when you upgrade a device software image or add a new NE model.
Visible in menu	Whether users can see the command in the Vision GUI client. **Tip** Do not make commands visible until you have tested them.
Description	Command description. A best practice is to fill this field because it will make it easier to identify user-created commands from those that are supplied with Prime Network.
Authorization	User access level required to run the command. (The user will be permitted to execute the command only if the device is in their scope.)

Step 4 Click **Next** to begin entering the command's implementation parameters.

Defining Execution Parameters for a Command

Note If you are creating a command that requires no input—for example, a simple show command—skip this section and proceed to Defining the Implementations (Scripts) for a Command, page 4-12.

The Execution Parameters determine the structure and format of the input form and the data that users must supply. Each parameter has an associated input type which determines how information is entered (for example, typing in a number or string, checking a check box, choosing from a drop-down list). You can specify:

- Required parameters and optional parameters.
- Parameters with default values which can be visible or hidden in the input form.

A command can have one parameter, multiple parameters, and multiple input pages (tabs) with their own parameters.

Step 1 Define your first parameter by clicking **New Parameter** in the Execution Parameters area, and provide the basic information for the execution parameter.

Execution Parameters Field	**Description**
Title	The name of the parameter. This is what will be displayed in the command dialog when the user runs the command.
Variable Name	Parameter name. This entry must be unique and can contain only letters, numbers, hyphens (-), and underscores (_).
Tooltip	A tooltip (maximum of 256 characters).

Step 2 Define the input type for the parameter.

Tip Use TextArea for multi-line parameter values (e.g., license, banner, and so on). Do not use it to configure multiple commands in a single parameter. Doing so will download all commands to the device, but the command output will only be partial.

.

For this type of user input:	**Choose this type:**
A type conversion	String
	Long
A number	Integer or float
Multi-line string that will be referenced as a single attribute (for example, information that can be cut and pasted (including end-of-line characters <CR><LF>))	TextArea
An IP address or IP subnet mask	IPSubnet
	IPAddress
A drop-down list of choices (see the following instructions)	Combo

If you choose **Combo**, an Add button appears in the wizard. Click **Add** to create choices for the combo box.

Add Combo Options	**Description**	**Example**	
Value	A value that will be associated with the option; for example, **1**.	**1**	**2**
Label	The description of the entry that is displayed in the selection list (drop-down list) of the input form, such as **Up**.	**Up**	**Down**

The example in the previous table would create a combo box with the choices Up and Down.

Step 3 Define the rest of the execution parameter. The choices depend on what you specified in the Type field.

Execution Parameters Field	Description
default	The default value (a number, enabled, etc.). To force the user to enter a value (without specifying a default), leave it blank and choose Required.
Display width	The maximum width in characters.
Visible	Whether the parameter will be visible in the command dialog. If you uncheck the box, the parameter is hidden from the user but can still be used in the command with a default value. To provide a constant value but keep it hidden from the user, uncheck the visible check box and provide a default value in the Default area.
Required	Specifies the input as mandatory. Mandatory input is designated by a red asterisk in the command.
Group	Indicates that the parameter belongs to a specified group. Parameters in a group are displayed together inside a box, with the group name as their title. For example, you could create a group called Authentication with two parameters: Username and Password. Username and Password would be enclosed in a box entitled Authentication.

Step 4 If you want the input to be either validated or populated when the user runs the command, click the **Validate** tab.

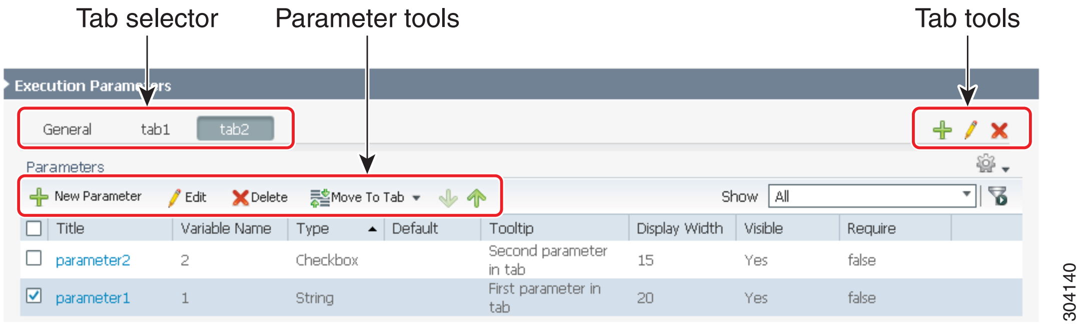
- On Populate—When Run Command is clicked, the BeanShell script should populate the parameters values according the script logic.
- On Validate—When Preview or Execute Now is clicked, the BeanShell script should validate the parameters values according the script logic.

Note When creating or editing On Populate or On Validate BeanShell scripts, you must restart the Vision GUI client for the changes to take effect.

Step 5 Click **OK**, and the new parameter is added to the list of Execution Parameters.

Step 6 Create more parameters as needed by repeating Step 1 through Step 5.

Step 7 If you want to create additional pages (tabs) or adjust the order of parameters, use the tools illustrated in the following figure.



☐	Title	Variable Name	Type	Default	Tooltip	Display Width	Visible	Require
☐	parameter2	2	Checkbox		Second parameter in tab	15	Yes	false
☑	parameter1	1	String		First parameter in tab	20	Yes	false

Step 8 When you have finished creating the parameters, click **Next** to begin entering the Command Implementations (the scripts that will be executed and the devices they can run on).

Defining the Implementations (Scripts) for a Command

Note All scripts are run at the network element level (IManagedElement).

The Command Implementations are the scripts that are run when the user executes the command. Implementations can use Bean Shell (Telnet or SNMP) or Cisco Prime Network Macro language (Telnet only). You can also specify a rollback script and the conditions that indicate the command has failed.

When you create an implementation you also define which device types the command applies to. You can choose different degrees of granularity:

- All devices
- Device family (such as routers)
- Device series (such as Cisco 7200 routers)
- Device model (Cisco 7207)

However, if you choose all devices, you cannot specify multiple software versions.

When a command is deployed, it overrides any inherited command from a higher level and automatically applies to elements below it. For example, the following tables shows how command versions can be overwritten:

	Results in these devices using this version:		
Chronological Actions	**7200 routers**	**7207 router**	**6509 switch**
1. Version 1 is applied to all devices (family)	1	1	1
2. Version 2 is applied to only the Cisco 7207 (model)	1	2	1
3. Version 3 is applied to all Cisco 7200 routers (series)	3	3	1
4. Version 4 is applied to all switches (family)	3	3	4

Step 1 Define the scripts that will run by clicking **New Implementation** in the Implementations area.

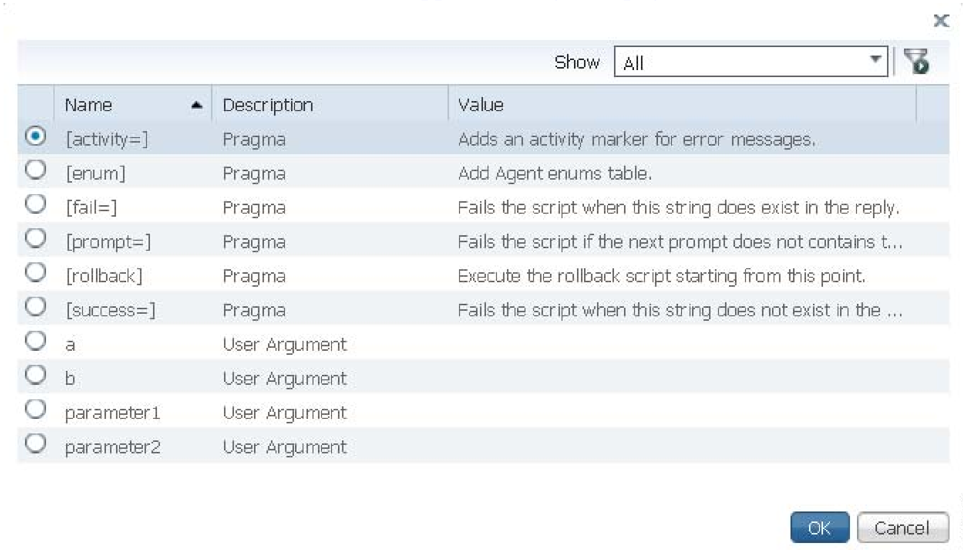
Step 2 In the General tab, provide the following script information.

Command Implementations Field	Description
Name	A meaningful name for the implementation, such as a device type, device series, or technology. For example, one profile could be *command*-AllDevices, another could be *command*-Cisco7600. **Note** A best practice is to associate the profile name with the devices you are choosing.
Device Types	The NEs that this command can be run on.
Software Version	The device software versions that are available for the chosen devices. If you select a device series or a non-Cisco device, this choice is disabled. If you subsequently update the software version on a device, you must also update the command.
Type	The scripting language to be used for the command: • CiscoCisco Prime Network Macro—Simple mode. (See Prime Network Macro Language, page C-1.) • BeanShell—Programmable mode; full scripting language. (See BeanShell Commands, page C-11.)
Protocol	The protocol to use. If the command has multiple profiles, they must use the same protocol. • Telnet • SNMP (BeanShell only) • TL1 (BeanShell only) Currently TL1 protocol is applicable only for CPT devices. **Note** You can create commands using only those protocols that are enabled on the device type in the Vision GUI.
Timeout	The timeout value for the command, in milliseconds. If a timeout occurs, the command continues to run in the background and the result is displayed in the command execution result screen.

Step 3 Click the Script tab to define the command script, using these guidelines:

Tip A best practice is to create the script in a text editor, then cut and paste it into the Script area of the Script tab.

- To view all user-defined and built-in parameters in the Command Builder application, position the cursor in the Script or Rollback field and press **Ctrl-Spacebar**. A dialog box is displayed that lists all available arguments (containing both the user-defined input argument and the built-in properties of the IMO context). Also see Supported Pragmas, page C-4.



Show: All

	Name	Description	Value
●	[activity=]	Pragma	Adds an activity marker for error messages.
○	[enum]	Pragma	Add Agent enums table.
○	[fail=]	Pragma	Fails the script when this string does exist in the reply.
○	[prompt=]	Pragma	Fails the script if the next prompt does not contains t...
○	[rollback]	Pragma	Execute the rollback script starting from this point.
○	[success=]	Pragma	Fails the script when this string does not exist in the ...
○	a	User Argument	
○	b	User Argument	
○	parameter1	User Argument	
○	parameter2	User Argument	

304125

Select an entry from the list and then click **OK** to add it to the Script or Rollback field.

- Enclose pragmas with square brackets: […].
- To enter a required carriage return in the command line, enter the escape sequence **&cr**.
- It is possible to use multiple pragmas in a single line, in which case all pragmas are analyzed. If the same type of pragma is repeated, only the last one is used.

Field	Description
Script	The Telnet or script lines to be sent to the NE. The script lines can contain optional inline directives (pragmas) for finer granularity control. For more information about the supported pragmas, see Supported Pragmas, page C-4.
Rollback	(Optional) The rollback script that is used when the command fails. **Note** If the rollback script fails, no additional actions are performed.
Failure Condition	(Optional) A general failure condition that applies to all script lines. To specify a failure condition: **1.** Check the Failure Condition check box. **2.** In the Failure Condition field, enter the text that is to be looked for during script execution. If the specified text appears in the reply, the command is aborted.

Step 4 When you are finished with the implementation, click **OK**.

Step 5 Create more profiles as needed by repeating Step 2 through Step 4.

Step 6 Click **Create** when you are done. The newly created command is displayed in the command repository table. (It is not yet deployed to devices; that happens when you select **Run Command**.)

Step 7 Test the command locally before deploying it. See Testing the Command Locally Before Deploying It to Other NEs, page 4-35

Creating a Command Sequence

Users with Configurator privileges can create command sequences. Sequences can only be executed from the command repository (not from the Vision GUI client). You cannot specify different devices for the different commands in a sequence.

Step 1 Choose a package. If you need to create a new package, click the New Package icon on the upper left side of the window and provide a name and optional description.

Tip Do not use the Default package because it contains all commands provided by Prime Network. This makes it easier to your manage user-created commands.

Step 2 Click **New Sequence** to open the Create Command Sequence wizard.

Step 3 Configure the sequences's basic information in the Command Sequence Info area. Note that there is no authorization level field; only users with Configurator privileges can create and run sequences. There is also no menu field because sequences can only be run from the command repository, not from the Vision GUI client.

Command Info Field	Description
Command Sequence Title	The name of the sequence. This is what will be displayed in the repository.
Command Sequence Name	The sequence name used and saved in the database.
Version	A number which helps you manage multiple instances of the command. A best practice is to change this when you have a command deployed to specific devices, and you want to update it for those devices.
Description	Command description. A best practice is to fill this field because it will make it easier to identify user-created commands from those that are supplied with Prime Network.

Step 4 Add the commands to the sequence in the Define Sequence area by clicking **Add Commands**. A dialog opens that lists all device-level commands in the repository (from all packages, including the Default package).

Step 5 Select the commands you want to add to the sequence and click **Add**.

Step 6 Configure the commands in the sequence. (You cannot edit the command contents.)

- Ensure that each command is also configured with a rollback sequence in the Create Command Sequence area. You can add a rollback sequence from the command repository by clicking the **Add** link.
- Adjust the command order.

Step 7 Click **Create** when you are done. The newly-created command is displayed in the command repository and can be executed from the repository.

Step 8 Test the command locally. See Previewing a Command Script and Checking Command Job Results, page 4-16.

Previewing a Command Script and Checking Command Job Results

When you preview a command, you can see how it will appear to users. The preview feature allows you to see the expected script output without performing any real command executions.

Once you execute a command with **Create Execution Job** (at the end of the following procedure) Prime Network creates a command job and displays a dialog that contains a hyperlink to the job details. To check the job status for completed and schedule jobs, select **Command > Jobs** from the Change and Configuration Management menu.

Step 1 Select the command or sequence in the repository and click **Run Command**.

Step 2 In the Select Device page, choose one of the following options:

- By Devices—Choose this option to select the device(s) that you want to run the command.
- By Groups—Choose this option to select the device group(s) that you want to run the command. There must be at least one device added to a device group for the command to run on that group. If a device is added to multiple device groups that are selected, the command will run only once on that device. For information on how to set up a device group, see the Setting Up CCM Device Groups" section.

Step 3 From the command input forms that are generated, enter the required input.

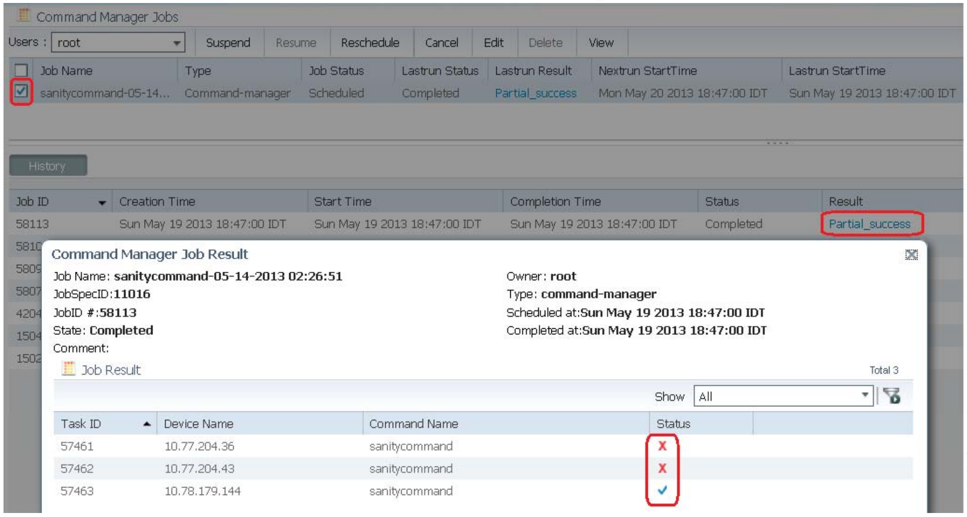
Step 4 To verify the parameters that you entered and see what the executed scripts will look like, click **Preview**.

Note Preview action is disabled for commands that use TL1 protocol.

Step 5 If the result is satisfactory, click **Create Execution Job** to:

- For a command, deploy it to the selected NEs.
- For a command sequence, save it to the repository.

Step 6 Click the job hyperlink to view the job results. To get job details, select the job and click the hyperlink under the Last Run Result column. If the job was for a command sequence, as in the following example, you can check the results of each command in the sequence.



☐	Job Name	Type	Job Status	Lastrun Status	Lastrun Result	Nextrun StartTime	Lastrun StartTime
☑	sanitycommand-05-14...	Command-manager	Scheduled	Completed	Partial_success	Mon May 20 2013 18:47:00 IDT	Sun May 19 2013 18:47:00 IDT

Job ID	Creation Time	Start Time	Completion Time	Status	Result
58113	Sun May 19 2013 18:47:00 IDT	Sun May 19 2013 18:47:00 IDT	Sun May 19 2013 18:47:00 IDT	Completed	Partial_success

Command Manager Job Result

Job Name: **sanitycommand-05-14-2013 02:26:51**
JobSpecID:**11016**
JobID #:**58113**
State: **Completed**
Comment:

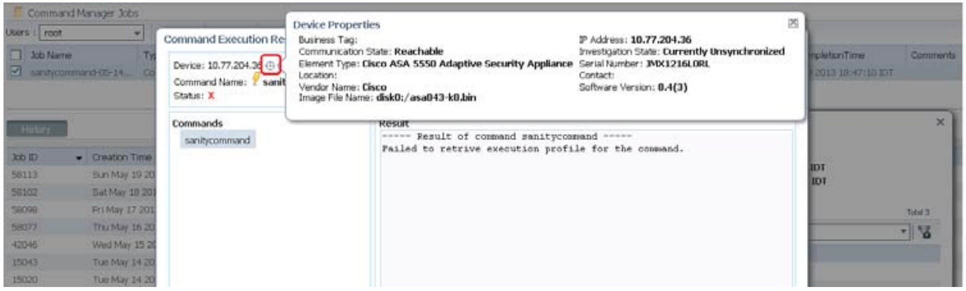
Owner: **root**
Type: **command-manager**
Scheduled at:**Sun May 19 2013 18:47:00 IDT**
Completed at:**Sun May 19 2013 18:47:00 IDT**

Job Result Total 3

Show All

Task ID	Device Name	Command Name	Status
57461	10.77.204.36	sanitycommand	X
57462	10.77.204.43	sanitycommand	X
57463	10.78.179.144	sanitycommand	✓

- If the job failed or was only partially successful, click the red X under Status to get details about the failure. By hovering over the Device Properties icon, you can get quick device details about the reachability of the device.



Device: 10.77.204.36
Command Name: sanit
Status: X

Device Properties

Business Tag:
Communication State: **Reachable**
Element Type: **Cisco ASA 5550 Adaptive Security Appliance**
Location:
Vendor Name: **Cisco**
Image File Name: **disk0:/asa043-k0.bin**

IP Address: **10.77.204.36**
Investigation State: **Currently Unsynchronized**
Serial Number: **JMX1216L0RL**
Contact:
Software Version: **0.4(3)**

```
----- Result of command sanitycommand -----
Failed to retrive execution profile for the command.
```

- If the job was successful, click the Transaction Log tab to see what scripts were sent to the device. Click the Debug Trace tab to view the information that was received from the device.

Enabling E-mail Notification for Command Scripts

Command Manager enables you to send automatic e-mail notifications about the status of jobs to users while executing the command or job scripts. To enable email notifications for the Command Manager, you need to add the SMTP host name in E-mail Settings pane of the Configurations Settings page in CCM.

Step 1 Choose **Commands** > **Command Repository**, select the command, and click **Run Command**.

Step 2 In the **Run Command** window, click the **Create Execution Job** pane, select the **Email Id(s)** checkbox, and enter the email id of the recipient, for example, user1@cisco.com. You can enter comma separated email ids for multiple recipients.

Step 3 Click **Create Job**. The e-mail is sent to the recipient with the following information:

- Job ID and Job spec ID
- Job name and type
- Job schedule time
- Job completion time
- Name of the device on which the job is executed
- Command name
- Status of the job executed

Step 4 Choose **Commands** > **Jobs**, select the job in the **Command Manager Jobs** page, and click **Edit**.

Step 5 In the **Edit Job** window, click **Create Execution** Job.

Step 6 In the **Email Id(s)** text box, enter the email ids of the recipient and click **Save**. You can enter comma separated email ids for multiple recipients. The e-mail is sent to the recipient with the following information:

- Job ID and Job spec ID
- Job name and type
- Job schedule time
- Job completion time
- Name of the device on which the job is executed
- Command name
- Status of the job executed

Running Commands Using Input Stored in a File

For bulk configurations, you can create an XLS or CSV files with parameter values and import the file into the command.

Step 1 Create a file that contains the parameter values.

a. Select the command from the repository and click **Run Command**.

b. Select the devices you want to configure and click the **Export** icon at the top right of the screen.

c. In the Export Execution Parameters dialog, click **OK** to continue with the export operation.

d. When your browser displays its file management dialog, download the file. The file is created with the name *command***.csv**.

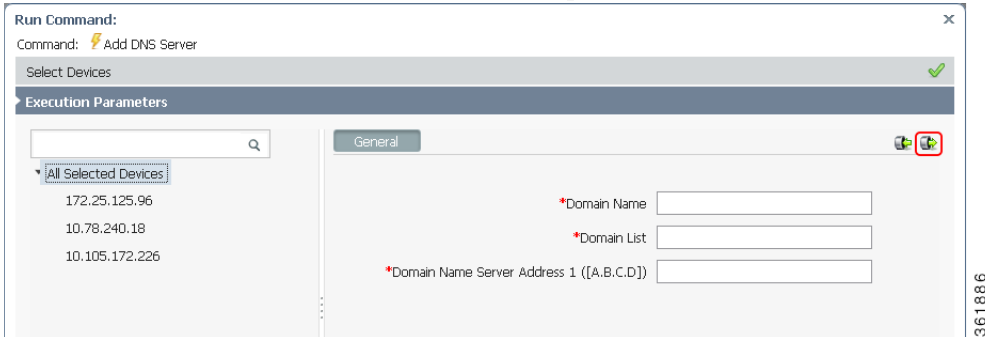
Step 2 Populate the *command*.csv file with your desired attribute values. One column is dedicated to each attribute.

Step 3 Run the command using your *command*.csv file as input.

a. Select the command in the repository and click **Run Command**.

b. Select the devices you want to configure. They must match the devices in your spreadsheet.

c. In the Execution Parameters area, click the import icon.



Run Command:

Command: Add DNS Server

172.25.125.96

10.78.240.18

10.105.172.226

*Domain Name

*Domain List

*Domain Name Server Address 1 ([A.B.C.D])

d. Navigate to the file that contains your input, and click **OK**.

e. In the Command Execution Job area, specify when the job needs to be executed (on daily, weekly, monthly, or on a specif time interval) and the e-mail ID(s) to which to send a notification after the scheduled job is complete. Click **Create Job**.

Step 4 Run the command.

Controlling Command Manager Jobs

Users with Configurator privileges can perform all job actions from the **Command Manager Jobs** window. This includes cancelling jobs, suspending jobs, rescheduling jobs, and editing job properties (such as changing the selected devices). Other users can only manage jobs that they own.

Command Manager jobs are automatically deleted according to the Prime Network job purging settings. By default, these jobs are never purged.

For examples of how to view job results, see Previewing a Command Script and Checking Command Job Results, page 4-16.

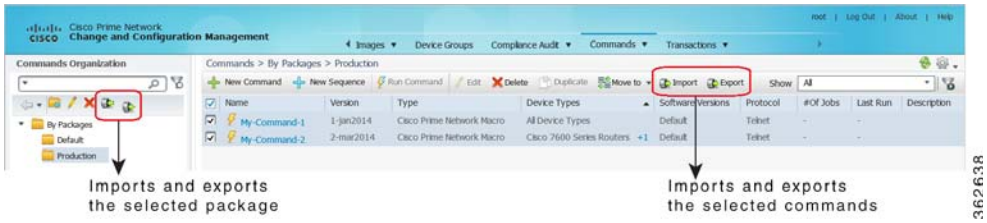
Moving, Importing, and Exporting Commands and Packages

Command Manager allows you to move, export, and import user-created commands. User-created commands are any commands that do not reside in the Default repository (which contains all commands that were provided with the Prime Network product).

To move user-created commands to a different package, select the commands in the repository, click the Move To icon, and choose the package.

You can also export and import user-created commands and packages. Exporting commands is useful when you want to clean up the repository or if you create commands that you plan to use at a later time. Figure 4-4 shows from where you should launch an import or export operation, depending on whether you are working with packages or individual commands.

Figure 4-4 Importing and Exporting Commands and Packages



☑	Name	Version	Type	Device Types	Software Versions	Protocol	#Of Jobs	Last Run	Description
☑	My-Command-1	1-jan2014	Cisco Prime Network Macro	All Device Types	Default	Telnet	-	-	
☑	My-Command-2	2-mar2014	Cisco Prime Network Macro	Cisco 7600 Series Routers +1	Default	Telnet	-	-	

Exported items, whether they are individual commands or a package of commands, are saved in a zip file on the client machine or on the gateway server. On the gateway server, commands are stored in *NETWORKHOME*/Command_Archive. If you do not supply a name in the export dialog box, Command Manager uses the following naming scheme:

Export ed Item	Default Name
Individual command	*command-name_date_time*
Group of commands	**Commands**_*date_time*
Package	*package-name_date_time*

Files you export to your local system are saved in your browser's download directory.

Import operations will query you for the location of the zip file that contains the commands or package. If you try to import a command that already exists in the command repository, you are prompted to overwrite the command or exit.

Cancelling Command Manager Jobs

To cancel the job that is in the running state:

Step 1 Select the job in the **Command Manager Jobs** window.

Step 2 Click **Cancel**.
The **Job Status** displays as Cancelled and the **Lastrun Status** displays as Cancelled with a hyperlink in the **Command Manager Jobs** window.

Step 3 Click the Cancelled hyperlink in the **Lastrun Status** field to display the **Command Manager Job Result** window.
The job result display the information about successful, unsuccessful, and cancelled tasks.

Deleting Commands and Packages

The following procedures explain how to:

- Remove a command from the Vision GUI client
- Remove a command or package from the command repository

Removing a Command from the Vision GUI Client

If you want to remove a user-created command from the Vision GUI client, you can hide it without removing it from the repository. You cannot hide out-of-the-box commands that are provided with Prime Network.

Note This operation may take some time depending on the number of NEs that are will be affected.

To remove a command from the Vision GUI client:

Step 1 Select the command in the repository and click the **Edit** icon.

Step 2 In the Command Info page, deselect the Visible in Menu check box.

Step 3 Click **Save**. The command is hidden in the Vision GUI client.

Deleting Commands and Packages from the Prime Network Command Repository

You can only delete user-created commands and packages. Out-of-the-box commands cannot be deleted. In addition, user-created commands can only be deleted if they are *not* part of:

- A scheduled job
- A command sequence

When a command is successfully deleted, it is removed from the repository and the Vision GUI client, and all related command data is deleted from the gateway and database.

When you delete a *package*, none of the user-created commands in the package are deleted. Instead, the commands are moved to the Default package, from where you can delete them.

To delete a command, make sure that you are working from the right package in the Commands Organization area. Then choose the command and click the Delete button above the commands table.

To delete a package, select the package and use the Delete button in the Commands Organization area.

Using Command Builder to Create Device Configuration Command Scripts

These topics explain how to use Command Builder to create commands and make them available to other NEs:

- What is Command Builder?, page 4-22
- Steps for Defining a New Command, page 4-22
- Command Builder Window, page 4-23
- Creating a Command Using Command Builder, page 4-24
- Testing and Publishing Commands, page 4-35
- Exporting Commands with Command Builder, page 4-38
- Importing Commands with Command Builder, page 4-38
- Deleting Commands, page 4-39

What is Command Builder?

Command Builder is an integral part of the Prime Network Vision GUI. It is used to create custom commands to be executed on specified devices. These commands can be tested and then published, which enables a wider scope of NEs to use them. You can publish commands to NEs of the same family, NE type, or software version.

Command Builder commands execute a programmable sequence of SNMP or Telnet command lines. Commands can be created using the BeanShell scripting language or the Prime Network Macro Language. Commands can be associated with any existing object group (IMO), type, or instance. This enables access to the live information of the network object with which the command is associated.

Only users with Administrator or Configurator privileges can create commands using Command Builder.

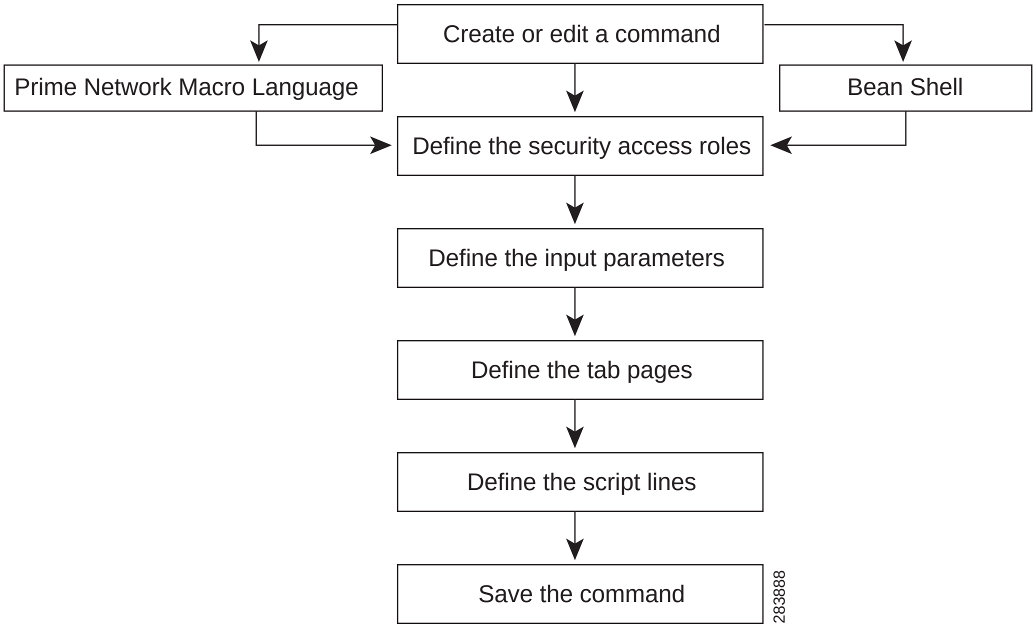
Commands can be used in a variety of ways:

- Using Command Builder, add the commands to the Prime Network Vision GUI client. Users will be able to run the commands by right-clicking an NE. When running the command, the user will be presented with a dialog for entering the input parameters that were defined during the command creation process.
- As building blocks in the creation of transactions (workflows) which can then be deployed to devices using Transaction Manager.
- Use the Prime Network API to integrate the commands into external configuration applications. For information on how to do this, see the *Cisco Prime Network Integration Developer Guide*.

Steps for Defining a New Command

These steps describe how to create a new command definition using Command Builder and the order in which the steps must be performed.

Figure 4-5 Steps for Defining a New Command



At any time after the command has been defined, it can be tested, executed, and published to a wider scope of managed elements and network elements.

For more information, see Creating a Command Using Command Builder, page 4-24.

Command Builder Window

Command Builder is launched by right-clicking a specific managed element or a selected object within a managed element and selecting **Management > Command Builder**.

The Command Builder window lists all existing commands that are available to the NE. (The technology-based commands that are packaged with Prime Network are not editable).

The commands that are listed depend on the NE type and the user's access privileges.

Figure 4-6 Command Builder Window

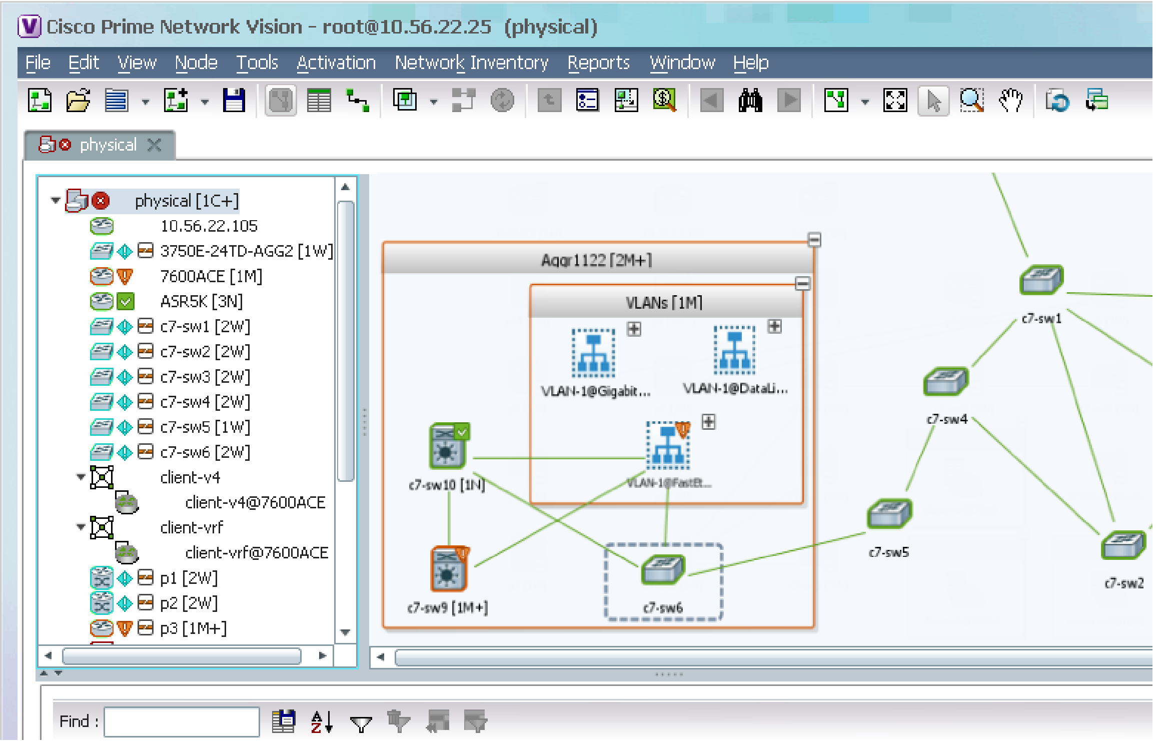


Table 4-1 Command Builder Window

Column	Description	Example:
Name	Name for the new command (also the filename).	ShowUsers
Menu Caption	The title that will appear in the command's GUI window when it is launched from an NE.	List of Current Users

Table 4-1 Command Builder Window (continued)

Column	Description	Example:
IMO Context	The inventory object associated with this command. **Note** Commands are associated with a selected object within a managed element, enabling the command to use the object properties in the command definition. For example, if you select a port object, the portAlias and port status properties are automatically made available to the command.	IManagedElement
Local	Specifies whether the command is inherited from a higher level (false) or is defined locally on the selected managed element (true).	false

Table 4-2 identifies the buttons that appear in the Command Builder window.

Table 4-2 Command Builder Window Icons

Button	Name	Description
	New Element	Creates a new command definition on the local client machine.
	Edit Element	Edits an existing command definition on the local client machine. When you edit an existing command, the changed command is saved as a local instance on the client machine. As with a new command, you can then publish it to make it available to other NEs. You can also edit commands that are packaged with Prime Network (for example, OAM commands that are launched from the right-click **Commands** menu). This is done by creating a clone of the command, which you can then edit.
	Delete Element	Deletes a command that exists on the local client machine (has not been published).
	Export Element	Saves a full command definition on the gateway.
	Import Element	Imports a command definition from the gateway to a local client machine. If the command already exists, it is overwritten (updated).
	Hierarchy Manager	Controls the NEs from which the command can be executed.
	Run Command	Previews and executes a command. After you create, modify, or update a command, we recommend that you wait a few seconds before executing it. If the gateway is busy, waiting a few seconds allows sufficient time for Command Builder to obtain the correct version from the registry.

Creating a Command Using Command Builder

When you create a new command, you first create a local instance of the command. After you have tested the command, you can publish it to make it available to other NEs.

To create a command:

Step 1 Launch Command Builder by right-clicking a managed element and choose **Management > Command Builder**. The window lists all existing commands that are available to the NE. (The technology-based commands that are packaged with Prime Network are not editable).

Step 2 Click the **New Element** icon in the toolbar to open the New Command dialog box.

Step 3 Enter the following information for your new command:

Field	Description
Name	Name for the new command (also the filename), such as **ShowUsers**.
Menu Caption	The title that will appear in the command's GUI window when it is launched from an NE, such as **List of Current Users**.
Menu Visible	Indicates whether or not the command should appear in the Prime Network Vision GUI client. For example, if you only want to use execute the command via the API, do not check the check box.
Menu Path	(If visible) The name that will appear in the NE's right-click menu (such as **Commands**).
Context IMO	The object associated with this command. The object is within a managed element. All subobjects that do not appear in the inventory tree (such as parameter groups of a port) are listed in a drop-down list.
Timeout	The timeout value for the command, in milliseconds. The default value is 120000 milliseconds (2 minutes). If a timeout occurs, the command continues to run in the background and the result is displayed on the Run Command screen.
Language	The scripting language to be used for the command: • Prime Network Macro—Simple mode. (See Prime Network Macro Language, page C-1.) • BeanShell—Programmable mode; full scripting language. (See BeanShell Commands, page C-11.)
Protocol	The protocol to use: • Telnet • SNMP (BeanShell only) • TL1 (BeanShell only)
Warning Visible check box	This check box and its accompanying message text field appear only if the system is set to generate a message upon command execution warning the user that the command could potentially interrupt network services. You can deselect this check box to override this setting for the current command. When the user executes this command, no message will be shown.
Warning Text	Shows the message that will be displayed to users upon command execution. You can change the message as required. The text field is disabled if the Warning Visible check box is deselected

Step 4 Click **Next**. The Command Authorizations dialog box is displayed.

Step 5 Select the security access roles that are authorized to execute the command.

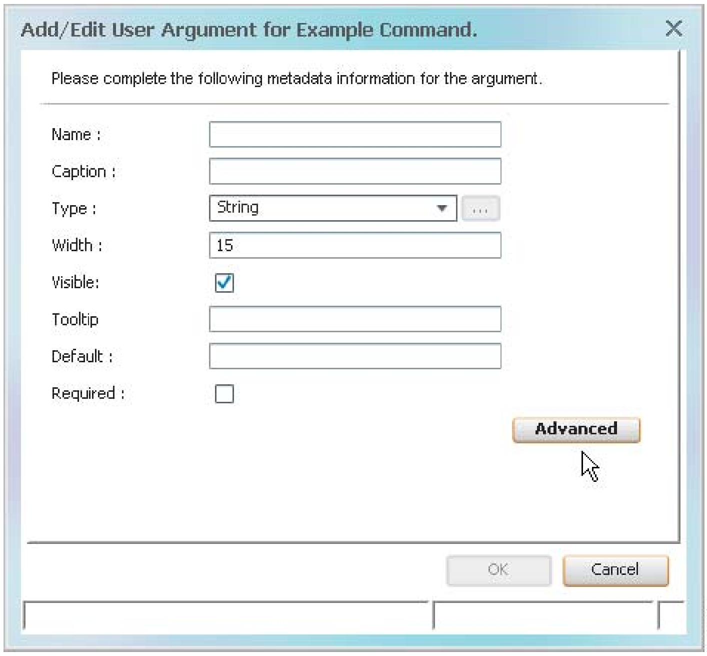
Step 6 Click **Next**. The User Input Arguments dialog box is displayed.

You can define any number of input parameters. These parameters determine the structure and format of the input form.The input form is automatically generated when the command is executed.

You can use either of two types of command parameters: built-in parameters and user-defined parameters, both of which are replaced in runtime. All parameters (both built-in and user-defined) are available during command editing via a selection list. For more information about Prime Network Macro Language, see Prime Network Macro Language, page C-1.

Step 7 To add a new argument, click **New**. The Add/Edit User Argument dialog box is displayed (Figure 4-7).

Figure 4-7 Add/Edit User Argument for Command Dialog Box



Add/Edit User Argument for Example Command.

Please complete the following metadata information for the argument.

Name :	
Caption :	
Type :	String
Width :	15
Visible:	☑
Tooltip	
Default :	
Required :	☐

Step 8 Enter the required information:

Property	Explanation
Name	Parameter name. This entry must be unique and can contain only letters, numbers, hyphens (-), and underscores (_).
Caption	Parameter display name. This entry is displayed in the Command Builder execution window.

Property	Explanation
Type	The type of input that will be required upon execution of the command. Only input values that are valid for the selected type are accepted. These values are validated during runtime. • String • Integer • IPSubnet • Combo—For more information about defining a Combo field type, see Defining a Combo Field Type for a Command Builder Command, page 4-30. • IP Address • Float • Long • Text Area—Multi-line string field type into which you can cut and paste information, including end of line characters <CR><LF>. This is referenced in the script as a single attribute. **Note** This feature should be used only for multi-line parameter values (e.g., license, banner, and so on). It is not intended for configuration of multiple commands in a single parameter. Doing so will download all commands to the device, but present only a partial output in the command builder output
Width	Field width, in number of characters. Relevant for the Command Builder execution window.
Visible	Indicates whether this parameter appears in the Command Builder execution window. Check this check box to display the parameter, or uncheck the check box to hide the parameter from the user. If the argument is hidden, it can still be used in the command (with its default value). **Note** When the parameter is not visible and has been assigned a default value, it can serve as a constant argument.
Tooltip	Specifies a tooltip for the command parameter. This string is displayed for the parameter field in the input form (see Testing the Command Locally Before Deploying It to Other NEs, page 4-35). The tooltip can be a maximum of 256 characters.
Default	A default value for the parameter.
Required	Indicates whether the argument is mandatory or optional. The mandatory arguments are displayed in bold font in the input form (see Testing the Command Locally Before Deploying It to Other NEs, page 4-35).

Click the Advanced tab to open the Add Argument Advanced Controller dialog box.

The Advanced option enables you to do additional actions on the parameter values using BeanShell.

- On Populate—The BeanShell script, executed when you click Run Command, can be used to populate the parameters values according the script logic.
- On Validate—The BeanShell script, executed when you click Preview or Execute Now, can be used to validate the parameters values according the script logic.

Note When creating or editing On Populate or On Validate BeanShell scripts, you must restart the client for the changes to take effect.

Step 9 Click **OK**. The newly created argument is displayed in the User Input Arguments dialog box.

Step 10 To change sequence in which arguments appear in the input form, select an argument and click **Move Up** or **Move Down**.

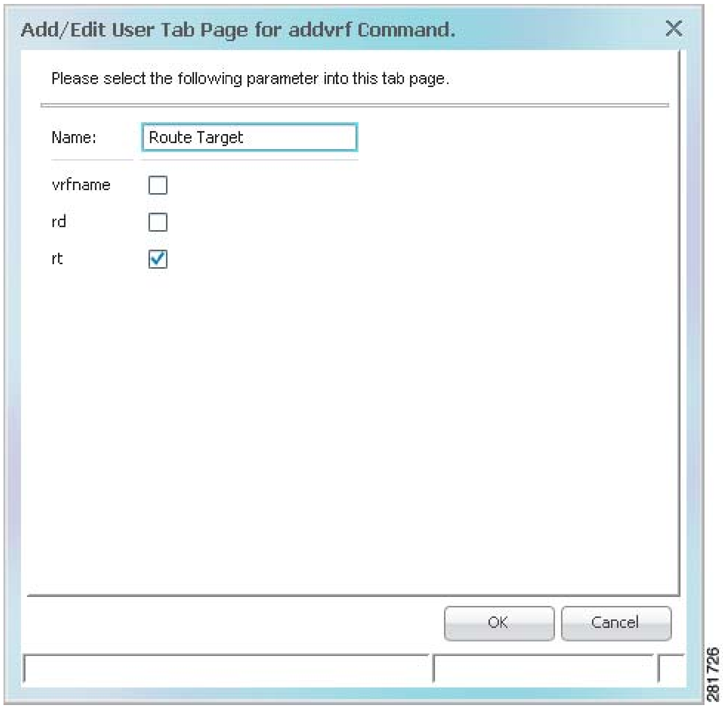
Step 11 Click **Next**. The Tab Pages dialog box is displayed.

Use this feature if you want your input form to have different tabs. By default, all of the parameters are displayed in a single tab, General.

Step 12 If you want to use tabs, do the following:

a. Click **New**. The Add/Edit User Tab Page dialog box is displayed.

Figure 4-8 Add/Edit User Tab Page for Command Dialog Box



Add/Edit User Tab Page for addvrf Command.

Please select the following parameter into this tab page.

Name: Route Target

vrfname □
rd □
rt ☑

b. Enter a name for the Tab and select the required parameter.

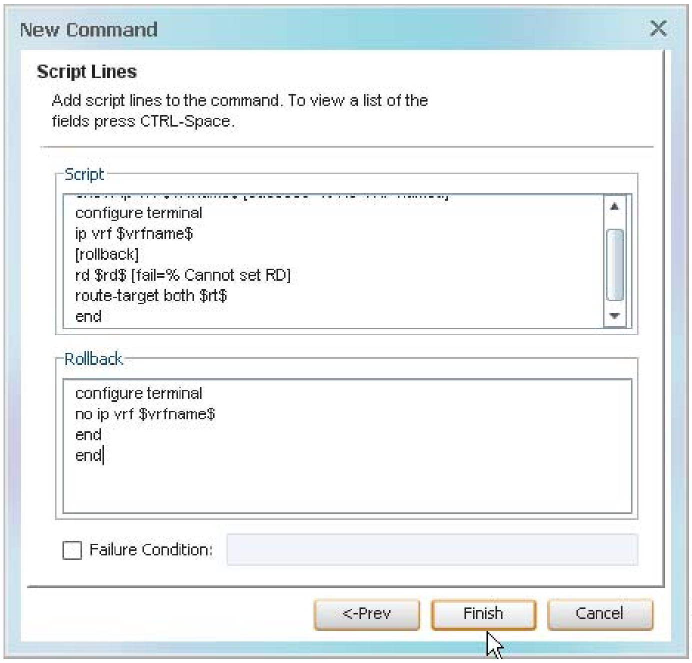
You can define a maximum of 20 tabs for a command. The name of the tab can contain a maximum of 20 characters.

c. Repeat the steps for additional tabs.

Step 13 Click **Next** to define the command script, which can be either Prime Network Macro Language scripts or Beanshell scripts. For more information, see Command Manager and Command Builder: Macro Language and Beanshell Reference, page C-1. The Script Lines dialog box is displayed, enabling you to add or edit a script line, as shown in Figure 4-9.

Note An efficient way to do this is to create the script in a text editor, then cut and paste it into the New Command dialog box.

Figure 4-9 Script Lines Dialog Box


New Command
Script Lines
Add script lines to the command. To view a list of the fields press CTRL-Space.
Script

```
configure terminal
ip vrf $vrfname$
[rollback]
rd $rd$ [fail=% Cannot set RD]
route-target both $rt$
end
```

```
configure terminal
no ip vrf $vrfname$
end
end
```

Step 14 Enter the required information, using the following guidelines:

- To view all user-defined and built-in parameters in the Command Builder application, position the cursor in the Script or Rollback field and press **Ctrl-Spacebar**. A dialog box is displayed that lists all available arguments (containing both the user-defined input argument and the built-in properties of the IMO context). Select an entry from the list and then click **OK** to add it to the Script or Rollback field.
- Pragmas are enclosed with square brackets: […].
- It is possible to use multiple pragmas in a single line, in which case all pragmas are analyzed. If the same type of pragma is repeated, only the last one is used.
- If carriage returns are required in the command line, enter the escape sequence **&cr**.

Field	Description
Script	The actual Telnet script lines sent to the NE. The script lines can contain optional inline directives (pragmas) for finer granularity control. For more information about the supported pragmas, see Supported Pragmas, page C-4.
Rollback	(Optional) The rollback script that is used when the command fails. **Note** If the rollback script fails, no additional actions are performed.
Failure Condition	(Optional) A general failure condition that applies to all script lines. To specify a failure condition: **1.** Check the Failure Condition check box. **2.** In the Failure Condition field, enter the text that is to be looked for during script execution. If the specified text appears in the reply, the command is aborted.

Step 15 Click **Finish**. The Create Command dialog box is displayed.

A LED indicates the progress or status of the command as it is being saved to the registry:

- Blue—The command definition is being saved.
- Green—The command has been created or updated successfully.
- Red—Command Builder failed to create or update the command.

Step 16 Click **Close** when the command has been successfully saved. The newly created command is displayed in the Command Builder table.

See Testing the Command Locally Before Deploying It to Other NEs, page 4-35 to test the command.

See Publishing Commands to Other NEs, page 4-35 to publish the command.

Note If you plan to publish the command to multiple network elements so you can work on commands in bulk, be sure to delete your local copy of the command after it is published. Otherwise, when you choose multiple managed elements, the command will not be listed.

Defining a Combo Field Type for a Command Builder Command

If you want to use a combo box in your input form, choose **Combo** in the Add/Edit User Argument dialog box (Step 8 of the previous procedure), a Browse button is enabled. Use it to create a selection list (drop-down list) of valid options that will be displayed.

This example creates a combo box with Up and Down choices.

Step 1 Select **Combo** in the Type field of the Add/Edit User Argument for x Command dialog box.

Step 2 Click **Browse**. The Selection List dialog box is displayed.

Step 3 Enter the required information:

Field	Description
Value	A value that will be associated with the option; for example, **1**.
Label	The description of the entry that is displayed in the selection list (drop-down list) of the input form, such as **Up**.

Step 4 Click **Add**.

Step 5 Repeat Steps 3 and 4 as needed until you have added all entries (such as **2** and **Down**).

Step 6 Click **Close**.

End-to-End Example of Creating a Macro Language Command

This topic provides an example of how to create the command, **addvrf**, using Prime Network Macro Language.

Step 1 Select the element that you want to configure this command for in the Prime Network Vision tree pane, context pane, or Inventory window.

Step 2 Right-click the element, then choose **Management > Command Builder**. The Command Builder wizard is displayed.

Step 3 Click **New**. The New Command dialog box is displayed.

Step 4 Define the command identification information:

- Name—addvrf
- Caption—Add VRF
- Menu Visible—Checked
- Menu Path—VRF Commands
- Context IMO—Automatically displayed
- Timeout—180000
- Language—Prime Network Macro
- Protocol—Telnet is selected by default

Step 5 Click **Next**. The Command Authorizations dialog box is displayed.

Step 6 Select the security access role Administrator.

Step 7 Click **Next**. The User Input Arguments dialog box is displayed.

Step 8 Click **New**. The Add/Edit User Argument for addvrf Command dialog box is displayed.

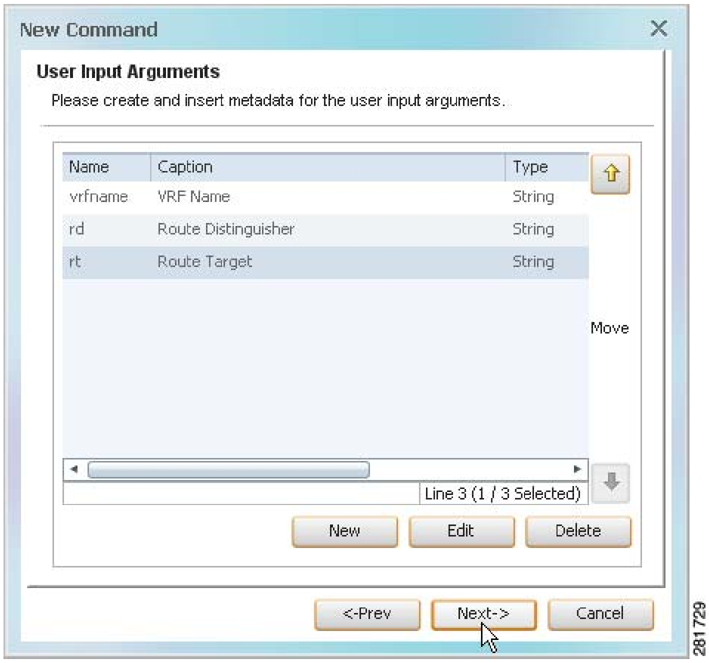
Step 9 Using Table 4-3, enter the information for the parameters.

Table 4-3 addvrf Command Parameters

Field	Parameter 1	Parameter 2	Parameter 3
Name	vrfname	rd	rt
Caption	VRF Name	Route Distinguisher	Route Target Both
Type	String	String	String
Width	15	15	15
Visible	Select	Select	Select
Tooltip	Enter VRF name	Enter Route	Enter Target
Required	Checked	Unchecked	Checked

Step 10 Click **OK** in the Add/Edit User Argument dialog box after each parameter. The new user-defined input parameters are displayed in the User Input Arguments dialog box (Figure 4-10).

Figure 4-10 User Input Arguments Dialog Box



New Command

User Input Arguments

Please create and insert metadata for the user input arguments.

Name	Caption	Type
vrfname	VRF Name	String
rd	Route Distinguisher	String
rt	Route Target	String

Line 3 (1 / 3 Selected)

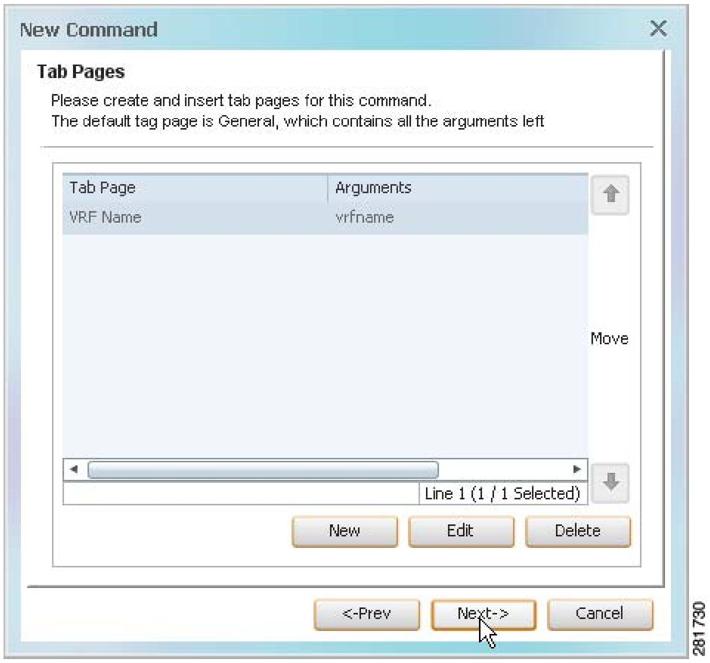
Step 11 Click **Next**. The Tab Pages dialog box is displayed.

Step 12 Click **New**. The Add/Edit User Tab Page dialog box is displayed.

Step 13 Enter a VRF Name and select **vrfname**.

Step 14 Click **OK**. The newly added tab pages are displayed in Tab Pages dialog box.

Figure 4-11 Tab Pages Dialog Box



New Command

Tab Pages

Please create and insert tab pages for this command.
The default tag page is General, which contains all the arguments left

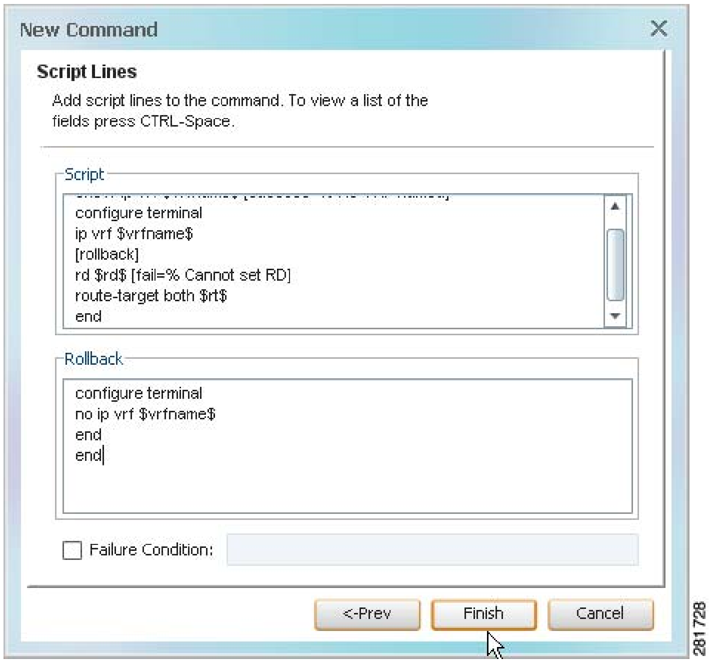
Tab Page	Arguments
VRF Name	vrfname

Line 1 (1 / 1 Selected)

Step 15 Click **Next**. The Script Lines dialog box is displayed, enabling you to add the script lines.

Step 16 Add the script lines, as shown in Figure 4-12.

Figure 4-12 Script Lines Example



Script Lines

Add script lines to the command. To view a list of the fields press CTRL-Space.

Script

```
configure terminal
ip vrf $vrfname$
[rollback]
rd $rd$ [fail=% Cannot set RD]
route-target both $rt$
end
```

Rollback

```
configure terminal
no ip vrf $vrfname$
end
end
```

□ Failure Condition:

Step 17 Click **Finish**. The Create Command dialog box is displayed. LEDs indicate the progress or status of the command as it is being saved to the registry.

Step 18 Click **Close** when the command is complete. The newly created **addvrf** command is displayed in the Command Builder table.

Step 19 Select the **addvrf** command in the Command Builder table.

Step 20 Run the command in one of the following ways:

- Click **Run Command** in the toolbar.
- Choose **Tools > Run Command**.
- Right-click the command, then choose **Run Command**.

The input form is generated and displayed. The Route Distinguisher and Route Target are displayed in General tab and vrfname is displayed in a separate tab called VRF Name.

In the General tab, the Route Target is a mandatory field (bold font) and Route Distinguisher is an optional field.

Step 21 Enter values in the VRF Name, Route Distinguisher, and Route Target fields.

Step 22 Click **Preview** to see how the command, including variables, looks before it is executed.

Step 23 Click **Execute Now** to view the results of the **addvrf** command as it is being executed.

Step 24 Close the input form to return to the Command Builder window.

Testing and Publishing Commands

The following topics describe how to test and publish commands:

- Testing the Command Locally Before Deploying It to Other NEs, page 4-35
- Publishing Commands to Other NEs, page 4-35

Testing the Command Locally Before Deploying It to Other NEs

Before you publish a command and apply it to other NEs, test it on your local NE using the following procedure. Once you have done this, you can apply it manually to multiple managed NEs, or publish it and make it available to a larger group of NEs (by family, NE type, or software version).

Step 1 From the Command Builder window, right-click the command and choose **Run Command**. Command Builder generates your input form.

Step 2 Complete the input form and click **Preview.**

Step 3 Click the **Result** tab to verify the command and variables before executing the command on the NE.

Step 4 If the command is successful and you want to execute it on the local NE, click **Execute Now**.

Note You might be prompted to enter your device access credentials. Once you have entered them, these credentials will be used for every subsequent execution of a command in the same GUI client session. If you want to change the credentials, click **Edit Credentials**. The Edit Credentials button will be disabled if the command is scheduled to run at a later time or if it is an SNMP command.

The queued commands can be terminated, if required, by clicking **Abort**.

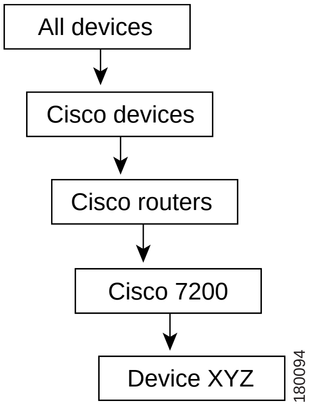
Alternatively, if you are authorized by the Administrator to schedule jobs, you can use the Scheduling tab to schedule the command to run at a later time. You can check the job results by selecting **Tools > Scheduled Jobs** from the Prime Network Vision Tools menu.

Every command that is executed is logged in the Prime Network event database. The command's execution history can be viewed using Prime Network Events. For more information about Prime Network Events, see the *Cisco Prime Network 4.2.2 User Guide*.

Publishing Commands to Other NEs

Publishing is the process of applying a command to a wider scope of managed NEs, so that the other NEs can use the new command. The Command Builder Hierarchy Manager dialog box controls the publishing of commands across the *inheritance hierarchy*. Figure 4-13 shows an example of an inheritance hierarchy. In this example, the top level of the hierarchy is All devices and the lowest level of the hierarchy is Device XYZ.

Figure 4-13 Inheritance Hierarchy Example



When a command is published to a node in the hierarchy, it overrides any inherited command from a higher level and automatically applies to all its children. For example, if a command is published to Cisco 7200, it overrides any variant of this command it inherited from a higher level and is assigned to all NEs of type Cisco 7200 in the system. NEs above the Cisco 7200 class are not affected.

Note A user with a Configurator role can add and publish commands on all NEs regardless of their assigned device scopes.

To publish a command:

Step 1 From the Command Builder window, right-click the command and choose **Hierarchy Manager**.

Note The names displayed in the VNE Hierarchy Location column depends on the registry settings. It is determined by the VNE schema (which controls the NEs that are managed and modeled by Prime Network). For example, a user-friendly name is displayed if a hierarchy path has been defined in the registry.

Rows are displayed in descending order with the top row representing the highest level of the hierarchy and the bottom row representing the lowest level of the hierarchy. The following information is provided for each level:

- Exist—If checked, indicates that a local variant of the command exists for that VNE hierarchy location.
- Registry Key—The hierarchy path as defined in the registry.

Table 4-4 describes the tools that are displayed in the Hierarchy Manager dialog box.

Table 4-4 Hierarchy Manager Dialog Box Tools

Button	Description
⎘	Copies the command from a selected node in the hierarchy so that it can be pasted onto another node in the hierarchy. A copy icon is displayed to the left of the selected node.
✂	Cuts the command from a selected node in the hierarchy so that it can be moved to another node in the hierarchy. A cut icon is displayed to the left of the selected node.
📋	Pastes the command that was copied or cut from a selected node in the hierarchy onto another node in the hierarchy. A paste icon is displayed to the left of the selected node.
🗑	Deletes the command from the selected node in the hierarchy. **Note** If the command has been deleted from all nodes, it is removed from the list in the Command Builder window.
💾	Saves a full command definition to a file that can later be imported to another managed element. For more information, see Exporting Commands with Command Builder, page 4-38.
📥	Performs one of the following actions, depending upon whether or not a version of the command definition already exists in Command Builder: • For new command definitions, imports a full command definition to a managed element. For more information, see Importing Commands with Command Builder, page 4-38. • For existing command definitions, replaces the existing command definition with an updated version of the command definition.

Step 2 Select the node in the hierarchy from where you want to Copy/Cut the command.

Step 3 Click **Copy** or **Cut** in the toolbar to copy or cut the command.

Step 4 Select the node in the hierarchy where you want to publish the command.

Step 5 Click **Paste** on the toolbar to paste the command. The command is published to the selected node in the hierarchy.

Note If you plan to publish the command to multiple network elements so you can perform commands in bulk, be sure to delete your local copy of the command after it is published. Otherwise when you choose multiple NEs, the command will not be listed.

If you select a group of network elements but the command is not listed, it is likely due to one of the following reasons:

- The command was not published to at least one of the network elements. You must publish the command to that network element using the hierarchy manager.
- One of the network elements has a local copy of the command. You must delete the local copy of the command using the hierarchy manager.

Note If a network element has more than one command instance, the command instance that is executed will be the one in the lowest hierarchy node.

Exporting Commands with Command Builder

When you export a command, a file containing the full command definition is created on the client machine. This is normally done when you want to manually import the command file to other NEs rather than use the Hierarchy Manager and the publishing procedure. The file is not copied to any other NEs; it is created so it can be imported by other NEs.

Step 1 In the Command Builder window, right-click the command and choose **Export Element**.

Step 2 Command Builder lists all versions of the command. In other words, if you have a 7206 NE with one version, but all other 7200 NEs have a different version, Command Builder would display both versions because both have been saved on your client machine. Check the version you want to export and click **OK**.

Step 3 Browse to the directory where you want to save the command.

Step 4 In the File name field, enter a name and click **Save**. The command is saved in XML format on the local client machine.

Importing Commands with Command Builder

Importing commands makes new commands available to an NE. For example, if you created a new command for NE 1, you could export the command from NE 1, and then import it to NE 2. The export is performed by launching Command Builder from NE 1, and the import is done by launching Command Builder from NE 2, as long as NE 2 is the same NE type.

If the import operation will overwrite a command that already exists, one of the following will happen:

- For a local command, Prime Network overwrites the existing definition.
- For a published command, Prime Network generates an error message which states that the command already exists. You must delete the existing command instance using the procedure described in Deleting Commands, page 4-39.

You can import multiple commands using this procedure as long as all of the commands are associated with the same VNE.

To import commands:

Step 1 In the Command Builder window, choose **Tools > Import Element**.

Step 2 Browse to the directory that contains the commands that you want to import.

Step 3 Select the commands that you want to import. To select multiple commands, press **Shift** or **Ctrl** while choosing the commands.

Step 4 Click **Open**. The Import Elements dialog box is displayed.

If you select multiple files, Command Builder presents an Import Element dialog box for each command.

Step 5 In the Import Element dialog box, select the VNE hierarchy location for the specified command.

Step 6 Click **OK**. The Command Builder window is displayed.

Step 7 Click **Close**. The commands are imported and displayed in the Command Builder window.

Deleting Commands

The procedure for deleting commands depends on whether the command is local or has been published.

- Local commands can be deleted from the Command Builder window by right-clicking the command and choosing **Delete**.
- Published commands must first be deleted from the applicable nodes using the Hierarchy Manager (right-click the command, then choose **Hierarchy Manager**). You must delete all instances from all applicable nodes.

You can delete multiple commands at a time as long as all of the commands are associated with the same VNE. You may want to keep a local copy of the command before you delete it.

CHAPTER 5

Deploying Activation Workflows to Devices Using Transaction Manager

The following topics explain how to use the Transaction Manager to deploy activation workflows (transactions) to devices:

- What is Transaction Manager?, page 5-1
- Setup Tasks for Transaction Manager, page 5-2
- Steps for Creating and Deploying Transactions, page 5-2
- Transaction Manager Window and Repository, page 5-3
- Creating Transactions Using the Prime Network XDE SDK, page 5-4
- Importing Transactions Into Prime Network, page 5-4
- Previewing and Executing Transactions, page 5-4
- Finding and Controlling Transaction Jobs, page 5-8
- Cancelling Transaction Manager Jobs, page 5-8

What is Transaction Manager?

Transaction Manager provides a GUI framework you can use to schedule and run *transactions* (workflows) that are created using the Prime Network XDE Eclipse SDK. You can also run transactions using the NBI.

Transactions are XDE procedures that contain tasks that are grouped together and specified into a flow, with certain sequences, branches, and failure policies (including rollback procedures). Transactions can include commands that you created using Command Manager or Command Builder. To include Command Builder commands, you need to perform the tasks outlined in the required extensions to the XDE Eclipse SDK (which has special features to work with Command Builder scripts).

Transaction rules, parameters, and implementation are stored on the gateway. Transactions (XDE procedures) are saved as archive files (.xar) and maintained at *XMP_HOME*/xmp_inventory/xde-home/packages/site. Jobs are also stored on the gateway and are purged according to the global settings in Prime Network.

Setup Tasks for Transaction Manager

Before you use Transaction Manager, check the Administration GUI client settings that controls whether users must enter their credentials before running transactions that include command scripts.

A user must have Administrator or Configurator privileges to run transactions. If per-user job authorization is enabled, Administrator and Configurator users must also be granted job privileges. The following table explains this and other global settings that can affect the transactions function. (These are in the Administration GUI client under **Global Settings > Security Settings > User Account Settings**.

Administration GUI Client Global Setting	Description
Job Scheduling	Enables and disables the global per-user job authorization mode. If the mode is enabled, job scheduling privileges are controlled by the setting in the individual user accounts. • If this mode is enabled and an Administrator or Configurator user is granted privileges, they can schedule jobs across the product. • If this mode is enabled and an Administrator or Configurator user is not granted privileges, the job scheduling features in their GUI clients are disabled.
Execution of Commands	If enabled (that is, the "Ask for user credentials when running device configuration operation" check box is checked), users must enter their credentials when they execute a transaction. Provisioning and Audit events display an additional column that lists the username. This setting is disabled by default.

For more information on these global settings, see the *Cisco Prime Network 4.2.2 Administrator Guide*.

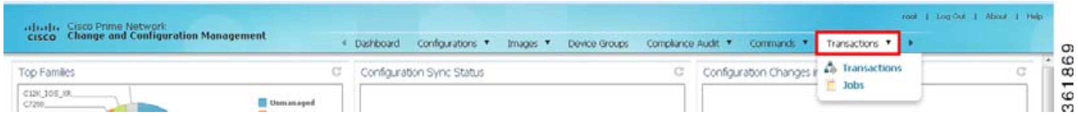
Steps for Creating and Deploying Transactions

The following table lists the steps you must follow to create and execute a transaction.

	Step Description	See:
Step 1	Check the settings that may affect who can run commands you create.	Setup Tasks for Transaction Manager, page 5-2
Step 2	Create the transaction using the Prime Network XDE Eclipse SDK.	Creating Transactions Using the Prime Network XDE SDK, page 5-4
Step 3	Import the transactions into Transaction Manager.	Importing Transactions Into Prime Network, page 5-4
Step 4	Preview the transaction script execution, and execute the transaction.	Previewing and Executing Transactions, page 5-4
Step 5	Check the transaction job status.	Finding and Controlling Transaction Jobs, page 5-8

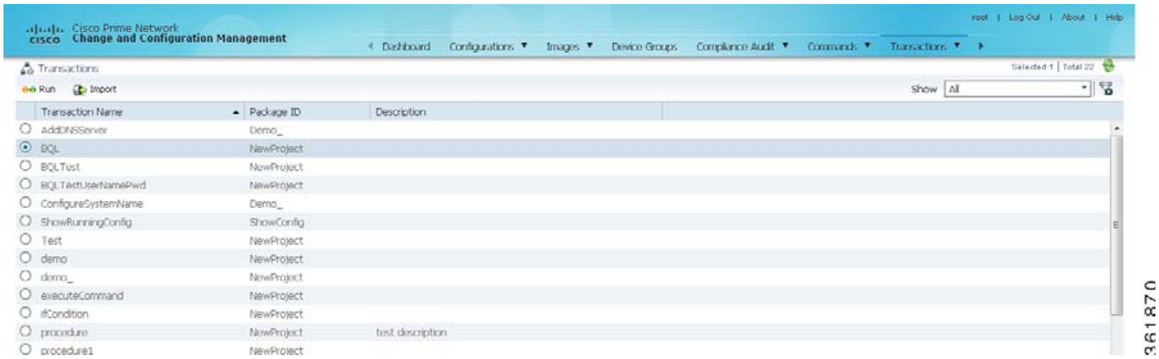
Transaction Manager Window and Repository

You can launch Transaction Manager from Change and Configuration Management, as shown in the following figure. (You may have to expand the title bar by clicking the arrow at the top left.)



When you select **Transactions > Transactions**, Prime Network displays all of the transactions on the Prime Network gateway. After you create transactions, you can import them into Transaction Manager and they will be listed in the transaction repository.

Figure 5-1 Transaction Manager Repository



Show: All

	Transaction Name	Package ID	Description
○	AddDNSServer	Demo_	
●	BQL	NewProject	
○	BQLTest	NewProject	
○	BQLTestUserNamePwd	NewProject	
○	ConfigureSystemName	Demo_	
○	ShowRunningConfig	ShowConfig	
○	Test	NewProject	
○	demo	NewProject	
○	demo_	NewProject	
○	executeCommand	NewProject	
○	ifCondition	NewProject	
○	procedure	NewProject	test description
○	procedure1	NewProject	

For information on the user roles required to run transactions, see Setup Tasks for Transaction Manager, page 5-2.

Creating Transactions Using the Prime Network XDE SDK

Use the Prime Network XDE SDK to create procedures called *transactions* (activation workflows). The XDE SDK is an Eclipse-based plug-in that provides a complete development environment for creating flowchart and device interaction packages. The XDE SDK includes a graphical flowchart editor for creating procedures, and uses an expression language that is focused on string-based and XML-oriented data processing.

Transactions can include Command Manager commands, Command Builder commands, SNMP commands, and CLI commands—or a combination of the four. However, if you want to use Command Builder commands in a transaction, you must create an XDE Protocol Abstraction Layer (PAL) Action to use in transactions. For installing XDE and creating the necessary PAL action, contact Advanced Services.

Importing Transactions Into Prime Network

You can import user-created commands and packages, either from the local system or a gateway server. Import operations will query you for the location of the zip file that contains the transactions.

Step 1 Select **Transaction > Transactions**, and then click the **Import** icon at the top left of the screen.

Step 2 Choose either **Local System** or **Gateway Server** to import the file, and then click **Browse**.

Step 3 Select the XDE package to be added to transaction repository. Only the files with a .xar extension is accepted.

Preview and execute the transaction as described in Previewing and Executing Transactions, page 5-4.

Previewing and Executing Transactions

When you preview a transaction, you can see how it will appear to users. Use the preview feature to see the Telnet and SNMP commands called by the transaction. This includes all Telnet and SNMP commands in Command Builder and Command Manager scripts. You can either choose a device or a device group on which you want to run the transaction.

After you execute a transaction with **Create Job**, Prime Network displays the job status page from which you can get job details.

Step 1 Choose **Transactions > Transactions**, select the transaction from the list, and then click the **Run** icon at the top left of the screen.

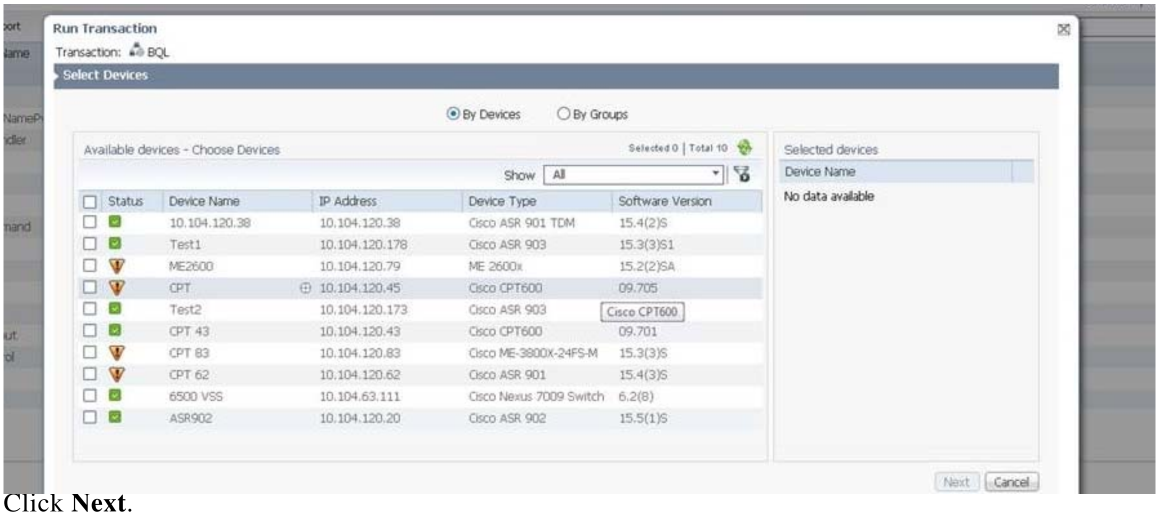
Step 2 In the Select Device pane, choose one of the following options:

- By Devices—Choose this option to select the device(s) on which you want to run the transaction.
- By Groups—Choose this option to select the device group(s) on which you want to run the transaction. There must be at least one device added to a device group for the transaction to run on that group. If a device is added to multiple device groups that are selected, the transaction will run only once on that device. For information on how to set up a device group, see the **Setting up CCM Device Groups** in *Cisco Prime Network 4.2.2 User Guide*.

Note The transaction will run on the devices that are available in the device group at the time of execution.

You can quickly check the device properties by clicking the icon next to the device IP address, as shown in Figure 5-2.

Figure 5-2 Device Properties in Transactions Manager



Run Transaction

Transaction: BQL

Select Devices

● By Devices ○ By Groups

Available devices - Choose Devices   Selected 0 | Total 10

Show: All

□	Status	Device Name	IP Address	Device Type	Software Version
□	🟩	10.104.120.38	10.104.120.38	Cisco ASR 901 TDM	15.4(2)S
□	🟩	Test1	10.104.120.178	Cisco ASR 903	15.3(3)S1
□	▽	ME2600	10.104.120.79	ME 2600x	15.2(2)SA
□	▽	CPT	⊕ 10.104.120.45	Cisco CPT600	09.705
□	🟩	Test2	10.104.120.173	Cisco ASR 903	
□	🟩	CPT 43	10.104.120.43	Cisco CPT600	09.701
□	▽	CPT 83	10.104.120.83	Cisco ME-3800X-24FS-M	15.3(3)S
□	▽	CPT 62	10.104.120.62	Cisco ASR 901	15.4(3)S
□	🟩	6500 VSS	10.104.63.111	Cisco Nexus 7009 Switch	6.2(8)
□	🟩	ASR902	10.104.120.20	Cisco ASR 902	15.5(1)S

Selected devices

Device Name
No data available

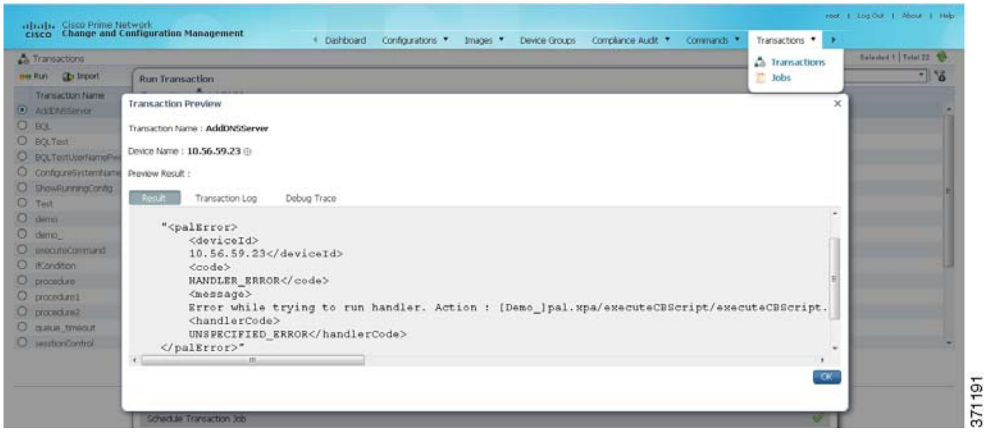
Step 3 Click **Next**.

Step 4 In the Transaction Input Parameters pane, enter the input parameters for the transaction, if required. (Not all transactions require input parameters.)

You can import or export input parameters for all or individual devices during transaction execution. For more information see, Running Transactions Using Input Stored in a File, page 5-7.

Click **Preview** to see the CLI commands called by the transaction.

Figure 5-3 Transaction Preview



Transaction Preview

Transaction Name : **AddDNSServer**

Device Name : **10.56.59.23**

Preview Result :

```
"<palError>
    <deviceId>
    10.56.59.23</deviceId>
    <code>
    HANDLER_ERROR</code>
    <message>
    Error while trying to run handler. Action : [Demo_]pal.xpa/executeCBScript/executeCBScript.
    <handlerCode>
    UNSPECIFIED_ERROR</handlerCode>
</palError>"
```

Step 5 Click **Next**. The Schedule Transaction Job pane is displayed.

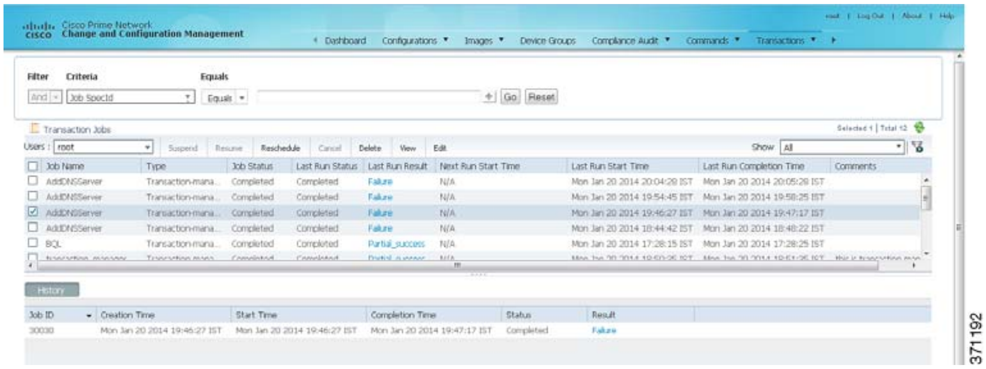
Step 6 In the Job Name field, enter a meaningful name for the transaction job. The default email ID is displayed in the E-Mail ID(s) text area. You can configure the default email ID in the Configuration Management Settings page (**Configurations** > **Settings**).

Step 7 Click **Create Job**.

Step 8 In the job dialog box, click the job hyperlink to view the status in the Transaction Jobs page.

Note An email with the job information is sent to the default email ID.

Figure 5-4 *Transaction Jobs*

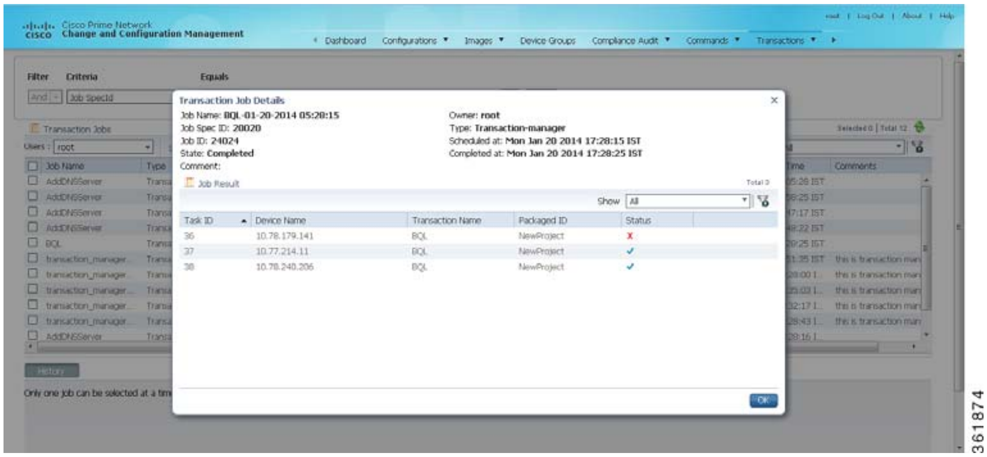


	Job Name	Type	Job Status	Last Run Status	Last Run Result	Next Run Start Time	Last Run Start Time	Last Run Completion Time	Comments
□	AddDNSServer	Transaction-mana...	Completed	Completed	Failure	N/A	Mon Jan 20 2014 20:04:29 IST	Mon Jan 20 2014 20:05:29 IST	
□	AddDNSServer	Transaction-mana...	Completed	Completed	Failure	N/A	Mon Jan 20 2014 19:54:45 IST	Mon Jan 20 2014 19:58:25 IST	
☑	AddDNSServer	Transaction-mana...	Completed	Completed	Failure	N/A	Mon Jan 20 2014 19:46:27 IST	Mon Jan 20 2014 19:47:17 IST	
□	AddDNSServer	Transaction-mana...	Completed	Completed	Failure	N/A	Mon Jan 20 2014 18:44:42 IST	Mon Jan 20 2014 18:48:22 IST	
□	BQL	Transaction-mana...	Completed	Completed	Partial_success	N/A	Mon Jan 20 2014 17:28:15 IST	Mon Jan 20 2014 17:28:25 IST	

Job ID	Creation Time	Start Time	Completion Time	Status	Result
30030	Mon Jan 20 2014 19:46:27 IST	Mon Jan 20 2014 19:46:27 IST	Mon Jan 20 2014 19:47:17 IST	Completed	Failure

Step 9 To view the job details, select the job and click the hyperlink under the Last Run Result column. The Transaction Job Details page is displayed with the job detail for that job instance.

Figure 5-5 *Transaction Job Details*



Transaction Job Details

Job Name: **BQL-01-20-2014 05:28:15**
Job Spec ID: **20020**
Job ID: **24024**
State: **Completed**
Comment:

Owner: **root**
Type: **Transaction-manager**
Scheduled at: **Mon Jan 20 2014 17:28:15 IST**
Completed at: **Mon Jan 20 2014 17:28:25 IST**

Task ID	Device Name	Transaction Name	Packaged ID	Status
36	10.78.179.141	BQL	NewProject	✗
37	10.77.214.11	BQL	NewProject	✓
38	10.78.240.206	BQL	NewProject	✓

You can check the results of each step in the transaction by clicking the hyperlink in the Status column.

Job Execution Results Tab	Provides Information About:
Result	The last step result of the transaction (provided by the transaction)

Job Execution Results Tab	Provides Information About:
Transaction Log	Log containing the results of device interaction (for example, the result of a command execution).
Debug Trace	Logging statements from the transaction.

Running Transactions Using Input Stored in a File

For bulk configurations, you can create an XLS or CSV files with parameter values and import the file into the transaction.

Step 1 Create a file that contains the parameter values.

a. Select the transaction from the repository, and then click **Run**.

b. Select the devices you want to configure, and then click the **Export** icon at the top right of the Transaction Input Parameters pane.

c. In the Export File dialog box, choose either **Export to Local System** or **Export to Gateway Server** to export the file.

Note If you choose the **Export to Gateway Server** option, the file name *transaction*.**csv is** populated in the Gateway Export Filename text box.

Step 2 Populate the *trsnasction*.csv file with your desired attribute values. One column is dedicated to each attribute.

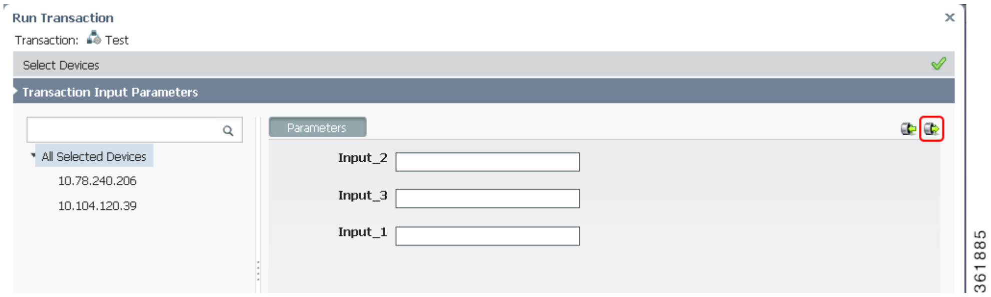
Step 3 Run the command using your *trsnasction*.csv file as input.

a. Select the command in the repository, and then click **Run**.

b. Select the devices you want to configure. They must match the devices in your spreadsheet.

c. In the Parameters area, click the **Import** icon.

Figure 5-6 Transaction Input Parameters



d. Choose **Local System** or **Gateway Server**, as required to navigate to the file that contains your input, and then click **OK**.

Step 4 Run the transaction.

Finding and Controlling Transaction Jobs

Transaction Manager jobs are automatically deleted according to the Prime Network job purging settings. By default, these jobs are never purged. For information on the user roles required to manage transaction jobs, see Setup Tasks for Transaction Manager, page 5-2.

For an example of how to get details about an executed job, see Previewing and Executing Transactions, page 5-4.

You can use the filter mechanism to locate executed and scheduled jobs using different search criteria, such as username, device name or IP, time stamps, and so forth. To view all of the possible criteria, click the Criteria drop-down list. When you choose a criteria, you then supply a value and click **Go**.

You can also locate transactions using a search based on input parameters. In this case, you supply the search criteria (in addition to the value), as shown in the following procedure.

Step 1 Choose **Transaction > Jobs**, select the transaction from the list, and click **Next**.

Step 2 From the Criteria drop-down list, choose **Input Parameters**. The Criteria field becomes an editable text box.

Step 3 In the Criteria text box, overwrite the *Input Parameter* text with any string; for example, **Interface**.

Step 4 Select an operator from the drop-down list; for example, **Contains**.

Step 5 Enter a value; for example, **gigabit/0/1**.

Step 6 Click **Go**.

Cancelling Transaction Manager Jobs

To cancel the job that is in the running state:

Step 1 Select the job in the **Transaction Manager Jobs** window.

Step 2 Click **Cancel**.
The **Job Status** displays as Cancelled and the **Lastrun Status** displays as Cancelled with a hyperlink.

Step 3 Click the Cancelled hyperlink in the **Lastrun Status** field to display the **Transaction Manager Job Result** window.
The job results display the information about successful, unsuccessful, and cancelled tasks.

PART 4

Extending and Customizing How Events and Faults are Managed

These topics explain how to extend and change fault management behavior in Prime Network:

- Adding Support for New Events Using the VCB, page 6-1
- Adding New Threshold-Crossing Alarms, page 7-1

CHAPTER 6

Adding Support for New Events Using the VCB

The VCB can be used to enable Prime Network to recognize additional events and to customize the way events are handled. For example, you might want Prime Network to recognize traps that are specific to a new technology or a custom syslog that you have defined. You might also want to change the settings of a system default event, for example, change the severity from major to minor.

In addition, you can define criteria for marking new or existing events as flapping events. Flapping events are events that rapidly change state and are identified based on the number of times and the frequency with which the change of state occurs. Prime Network does not send an event for each change of state but sends one event indicating that the event is flapping. Thereafter, no events are generated for any state change. The event is updated periodically if the flapping persists. A clearing event is sent when flapping has stopped.

You can also define expedite rules that will force polling whenever a specific event is received.

You can customize events using the VCB tool within Prime Network, or using VCB CLI commands. This chapter describes event customization using the Prime Network VCB GUI. For information on using the CLI method, see VCB CLI Reference: Events Commands, page B-31.

These topics explain how to use the VCB to create support for events, and to change how events are handled:

- Enabling Support for Unsupported Traps, page 6-1
- Enabling Support for a Custom Syslog, page 6-10
- Testing Your New Events, page 6-14
- Customizing Events, page 6-15

For general information about the VCB, see VCB Overview, page 3-1. For information on the level of support for various features in the different types of VNEs, see Comparison of Generic SNMP VNEs, U-VNEs, and Developed VNEs, page 3-3.

Enabling Support for Unsupported Traps

This procedure describes how to add unsupported traps as events in Prime Network based on a particular MIB definition file. You have the following options for adding support for traps:

- Add each trap individually. In this case, you create a single event with one or more traps of the same type (i.e., with the same prefix) as sub-types of the event.

- Add multiple traps simultaneously, in bulk. In this case, you upload multiple traps at once, irrespective of their type. An event is created for each trap. This is useful if you need to add support for a large number of traps of different types.

Adding Support for a Trap

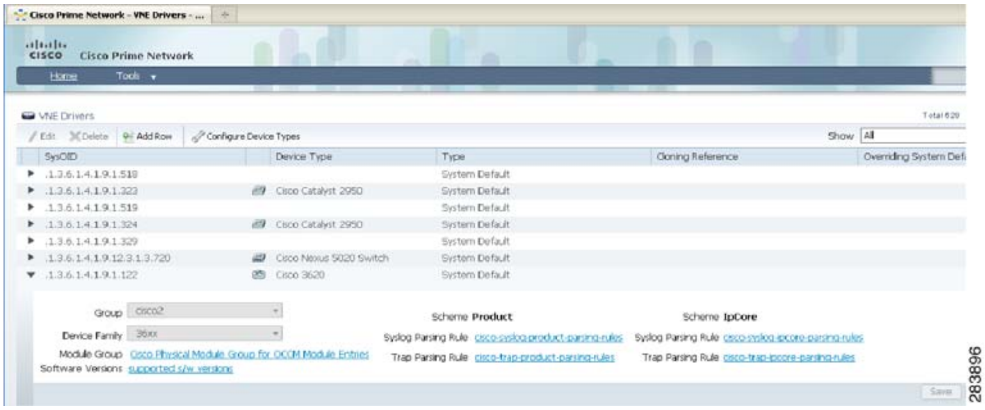
This procedure describes how to create a single event that represents a trap or a set of traps of the same type. If you want to add support for multiple traps of different types, see Adding Support for Multiple Traps Simultaneously, page 6-7.

Use the VCB to add support for an unsupported trap, as follows:

Step 1 In the VCB tool, go to the VNE Drivers tab.

Step 2 Click on the arrow next to the VNE driver on which you want the additional traps to be supported.

Figure 6-1 Expanded VNE Driver Properties Showing Parsing Rules



SysOID	Device Type	Type	Cloning Reference	Overriding System Def
.1.3.6.1.4.1.9.1.518		System Default		
.1.3.6.1.4.1.9.1.323	Cisco Catalyst 2950	System Default		
.1.3.6.1.4.1.9.1.519		System Default		
.1.3.6.1.4.1.9.1.324	Cisco Catalyst 2950	System Default		
.1.3.6.1.4.1.9.1.329		System Default		
.1.3.6.1.4.1.9.12.3.1.3.720	Cisco Nexus 5020 Switch	System Default		
.1.3.6.1.4.1.9.1.122	Cisco 3620	System Default		

Group: cisco2
Device Family: 36xx
Module Group: Cisco Physical Module Group for OCCM Module Entries
Software Versions: supported s/w versions

Scheme **Product**
Syslog Parsing Rule: cisco-syslog-product-parsing-rules
Trap Parsing Rule: cisco-trap-product-parsing-rules

Scheme **IpCore**
Syslog Parsing Rule: cisco-syslog-ipcore-parsing-rules
Trap Parsing Rule: cisco-trap-ipcore-parsing-rules

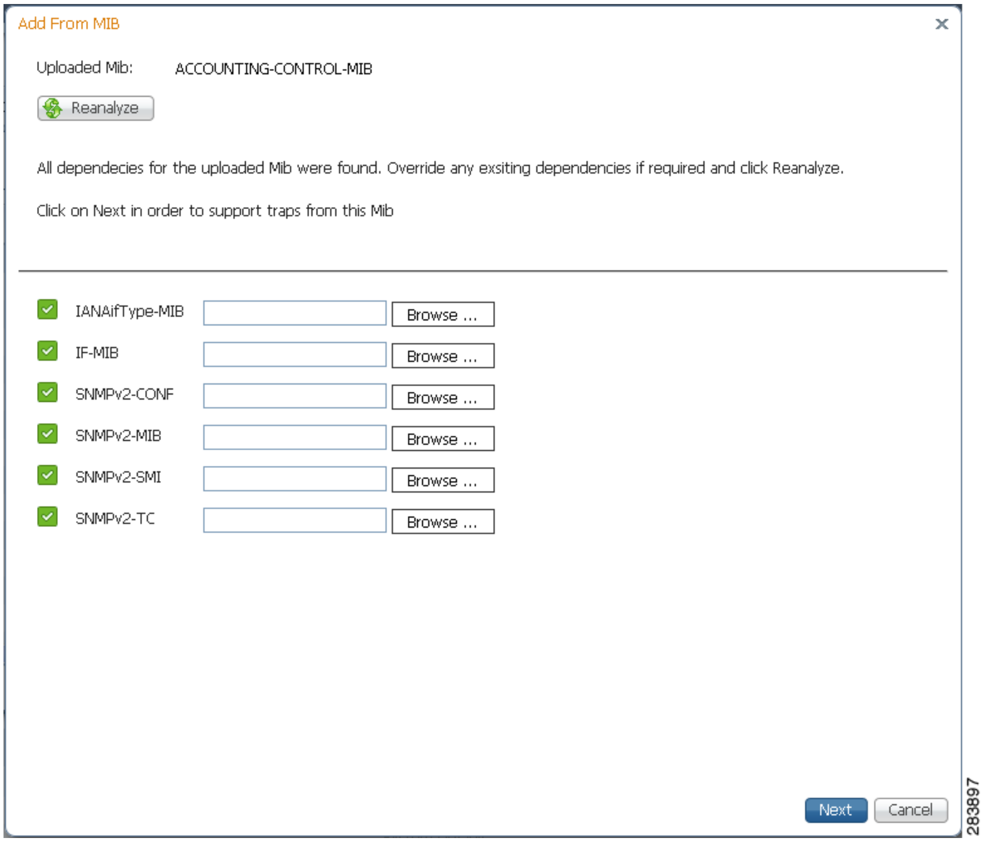
Step 3 Click the Trap Parsing Rule link to show a list of traps associated with this parsing rule.

Step 4 Click **Add from MIB**. This launches a wizard which enables you to analyze a specified MIB and select the traps to be supported.

Step 5 Click **Browse** and select the MIB file you want to upload to the Prime Network gateway. You can upload an individual MIB file or a zip file containing multiple MIB files. The file extension should be .mib or .my, .zip, or no extension. If you select a zip file, a list of the contained MIBs is displayed. You can select one or more of these MIBs to upload.

A list of MIB dependencies is displayed. A green check mark indicates that the dependency file has been found on the server.

Figure 6-2 MIB Dependencies



Add From MIB

Uploaded Mib: ACCOUNTING-CONTROL-MIB

All dependecies for the uploaded Mib were found. Override any exsiting dependencies if required and click Reanalyze.

Click on Next in order to support traps from this Mib

☑ IANAifType-MIB
☑ IF-MIB
☑ SNMPv2-CONF
☑ SNMPv2-MIB
☑ SNMPv2-SMI
☑ SNMPv2-TC

Step 6 If any of the dependencies is not found, click **Browse** and select the dependency file to upload.

Step 7 Click **Next**. A list of traps is displayed. A red icon to the left of the trap indicates that it is not supported. A green icon indicates that the trap is already supported.

Step 8 Check the check box next to the trap(s) to be supported. If you select multiple traps, they must be of the same type (i.e., they must have the same prefix). Click **Next**.

At this point, an event is created. Each trap you selected becomes a subtype of the new event. The Event Definition wizard is displayed to enable you to complete the definition of the new event.

Step 9 Click on **Step 1 - Event Definition** and provide the following information:

- Event name and OID—these are pre-populated but can be changed if necessary. The OID is the common prefix of the OIDs of the selected subtypes.
- Category—predefined category from 3GPP standards (according to ITU-T Recommendations X.733 and X.736). You can change the category if necessary.
- Nature—defines whether the event is automatically cleared by a clearing event or it needs to be manually cleared. Possible values are:
  - ADAC (Automatically Detected Automatically Cleared)—The event is automatically detected and automatically cleared by the system. For example, “link down” event. Select this option if the event has a clearing event, for example, “link up”.
  - ADMC (Automatically Detected Manually Cleared)—The event must be manually cleared by the user. For example, “DWDM fatal error” syslog. Select this option if the event does not have a clearing event.

Step 10 To define the criteria by which this trap will be considered a flapping event, check the Flapping check box. This would be relevant for network or communication related events that potentially could change state frequently. For example, link up/down, BGP neighbor up/down. When an event is flapping, Prime Network generates a single event indicating the flapping status and does not generate an additional event for each state change.
Enter the following information to define flapping criteria:

- Clear Interval—The flapping event will be cleared when the flapping has stopped for the specified time interval (in milliseconds).
- Flapping Interval and Flapping Threshold—To be defined as a flapping event, the event must change state every X milliseconds (Flapping Interval), and must occur Y consecutive times (Flapping Threshold). For example, if the Flapping Interval is 60000 msecs (one minute) and the Flapping Threshold is 5, the event must change state five times, every one minute, in order to be considered flapping.
- Update Interval and Update Threshold—An update event indicating that the event is still flapping will be sent either when the specified Update Interval has passed or when the event is received more times than the specified Update Threshold. For example, if the Update Interval is 100000 msecs and the Update Threshold is 500, an update event will be generated if the event remains flapping for longer than 100000 msecs or if the event is received more than 500 times, whichever occurs first.

Step 11 Click on **Step 2 - Subtype Definition**. You will see that a subtype has been created for each unsupported trap you selected. Edit the information for each subtype as required:

Field	Description
Name	Enter a unique name for the event subtype
Description to be Displayed	Enter a string that describes the event subtype. You can include dynamic values in the description string to provide more details. See Adding Parameter Values to Event Descriptions, page 6-19.
Severity	Select the severity to be attributed to the subtype.
Ticketable	Check the check box if you want Prime Network to create a ticket for this event if there is no root cause event to which it can be correlated. If you make a subtype ticketable, a ticket will be generated for it. When a non-ticketable subtype of the same event arrives (for example, a warning or clearing event), the ticket will be updated.
Auto-Clear	Check the check box if you want Prime Network to automatically clear the event. Prime Network clears a ticket if all of its events either are cleared or are configured for automatic clearing.
Correlate	Check the check box if you want the event to be correlated to a root-cause alarm.

In addition, if you selected the Flapping check box in the previous step, you will see that Flapping subtypes have been created. By default, the flapping subtype consists of the event name and the flapping subtype name. This can be changed as required; for example, you can also change the severity of the flapping subtype so that it matches the severity of the event.

Step 12 Click on **Step 3 - Subtype Identification**. In this step, you define how Prime Network will differentiate between the subtypes, as follows:

Field	Description
By TrapOID	Select this option if each subtype has a unique OID. In the Replacing Rules section, specify the OID suffix for each subtype. The OID suffix must be an integer.
By Varbind Value	Select this option if the subtypes have the same trap OID and you want to use one of the varbind values to differentiate between the subtypes. In the Replacing Rules section, select the required varbind from the drop-down menu or enter free text, and then define the values for each subtype.
By Varbind OID	Select this option if there is a varbind for each subtype. In the Replacing Rules section, specify the common prefix of the varbind OIDs and the suffix for each subtype.

Step 13 Click on **Step 4 - Association**, in which you associate the event with the VNE. Specify the following information:

Field	Description	
Source Type	The entity to which the event should be associated.	
	ManagedElement Key	Select this option if there is no specific interface or other component of the VNE from which the event is generated.
	Interface Key From Ifindex	Creates the interface device component key from the ifIndex and associates the event with the appropriate interface layer.
	Interface Key From Ifname	Associates the event with a specific interface that you specify in the Interface Identifier field.
	Logical Container Key	Associates the event with a designated logical container that you select in the Logical Container field.
	ModuleDC Key Given EntPhysicalIndex	Associates the event with a designated module DC.
	ModuleDC With SlotSubslot Value Key	Associates the event with the corresponding module, based on the slot number.
	Pw Interface Key From Tunnelindex	Associates trap events with the designated pseudowire tunnel interface.
Logical Container	Applicable only when the source type is Logical Container Key. This field lists the various logical containers for which the VCB supports event association. For example, BGP traps/syslogs can be associated with the MP-BGP type container, ISIS events with the ISIS System container, and so on.	
Instance Identifier Location	Specify whether the identifier of the event is based on a value or a varbind OID.	

Field	Description
Instance Identifier Varbind OID	Select the varbind that contains the instance information. Prime Network uses varbind OID to locate the varbind in the trap PDU and locates the instance information from either the OID or the value depending what was selected as the instance identifier location (OID or value).
Source Location	Specify whether the event source can be found in a value or a varbind OID
Source Varbind ID	Select the varbind that contains the source information. Prime Network uses varbind OID to locate the varbind in the trap PDU and locates the source information from either the OID or the value depending what was selected as the source location (OID or value).

Step 14 Optionally, click on **Step 5 - Expedite** if you want the device to be polled immediately for inventory updates upon receipt of the event, instead of waiting for the next polling cycle which may take up to twenty four hours for VNEs using reduced polling. Click **Add** to create an expedite rule and provide the following information:

Field	Description
Level	• Select VNE Component if the expedite command will run on a component of the VNE. • Select VNE if the expedite command will run on the VNE level.
Class	Shows the VNE-level or the component-level command classes. Select the command class that contains the relevant expedite command.
Command Name	Shows the commands belonging to the selected class. Select the relevant command.
Delay	Delay (in milliseconds). By default, the command will be executed immediately upon receipt of the event (default value is 0).
Key	Component level only. This is the entity by which the component will be identified. If the source type is the same as the source type specified for association, there is no need to specify the key again here.

Note Command expedition might result in unnecessary polling of devices which puts a load on the device and might affect performance. Therefore, before creating command expedition rules, consider whether the event is specific to a particular VNE or common to all VNEs and might affect the performance of VNEs using regular polling.

Step 15 Click on **Step 6 - Pattern** to determine which VNE drivers will be extended to support this event. This is determined by selecting the parsing rule groups per scheme that will be extended. The event will be supported on all VNE drivers that use the specified parsing rule groups.

Step 16 Click **Add** to select additional parsing rule groups, as required, either for the Product or IpCore scheme.

Note Certain parsing rules groups inherit from other groups. If you select multiple groups, make sure that your selection does not include a base (parent) group as well as the group that inherits from the base group. See Parsing Rule Group Inheritance Structure, page 6-13 for the relationship between parsing rule groups.

Step 17 Click **Finish**. A dialog box displays a list of the traps that were added.

Step 18 If you want to enable support for additional traps, click **Select Different Trap**, otherwise click **Close**.

Adding Support for Multiple Traps Simultaneously

This procedure describes how to add support for multiple traps at once. The traps do not have to be of the same type. This enables you to add support for traps in bulk, without having to add support for each trap individually. This procedure can be used if you want to add support for a large number of events that do not require complex processing, on the managed element level.

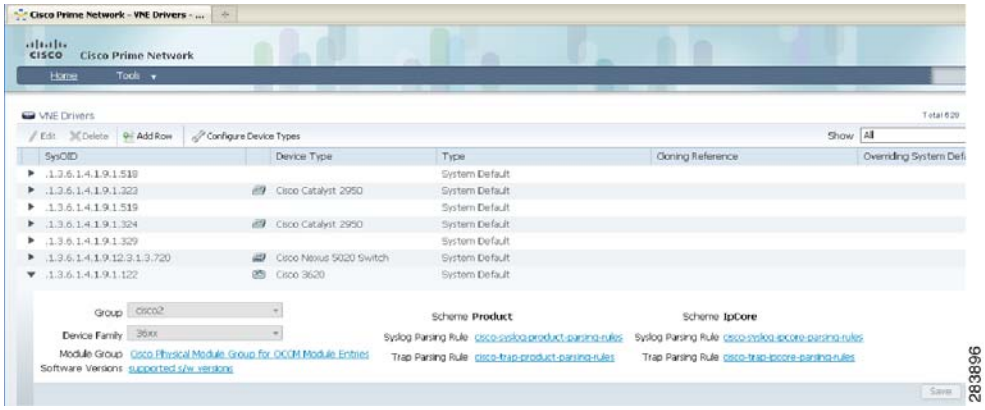
Note The traps added using this procedure will be associated at the managed element level only (not at the interface level or any other DC). In addition, you will not be able to expedite a command upon receiving events added using this method.

To add support for traps in bulk:

Step 1 In the VCB tool, go to the VNE Drivers tab.

Step 2 Click on the arrow next to the VNE driver on which you want the additional traps to be supported.

Figure 6-3 Expanded VNE Driver Properties Showing Parsing Rules



SysOID	Device Type	Type	Cloning Reference	Overriding System Def
.1.3.6.1.4.1.9.1.518		System Default		
.1.3.6.1.4.1.9.1.323	Cisco Catalyst 2950	System Default		
.1.3.6.1.4.1.9.1.519		System Default		
.1.3.6.1.4.1.9.1.324	Cisco Catalyst 2950	System Default		
.1.3.6.1.4.1.9.1.329		System Default		
.1.3.6.1.4.1.9.12.3.1.3.720	Cisco Nexus 5020 Switch	System Default		
.1.3.6.1.4.1.9.1.122	Cisco 3620	System Default		

Group: cisco2
Device Family: 36xx
Module Group: Cisco Physical Module Group for OCCM Module Entries
Software Versions: supported s/w versions

Scheme **Product**
Syslog Parsing Rule: cisco-syslog-product-parsing-rules
Trap Parsing Rule: cisco-trap-product-parsing-rules

Scheme **IpCore**
Syslog Parsing Rule: cisco-syslog-ipcore-parsing-rules
Trap Parsing Rule: cisco-trap-ipcore-parsing-rules

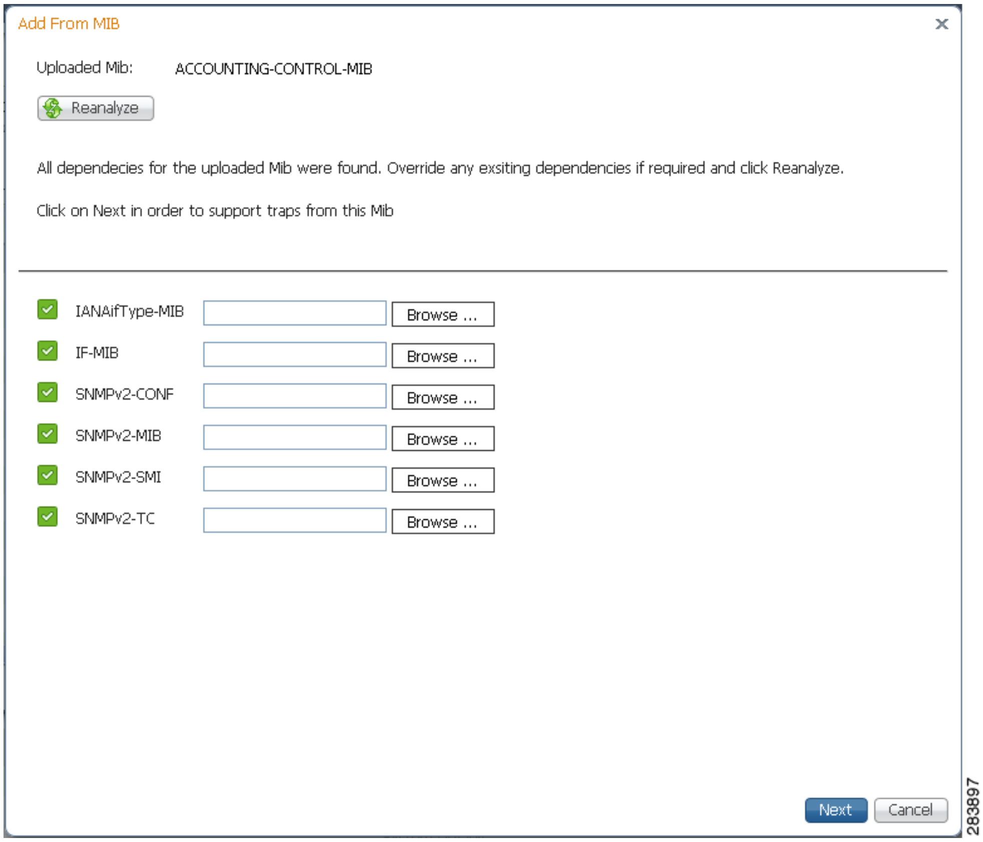
Step 3 Click the Trap Parsing Rule link to show a list of traps associated with this parsing rule.

Step 4 Click **Add from MIB**. This launches a wizard which enables you to analyze a specified MIB and select the traps to be supported.

Step 5 Click **Browse** and select the MIB file you want to upload to the Prime Network gateway. You can upload an individual MIB file or a zip file containing multiple MIB files. The file extension should be .mib or .my, .zip, or no extension. If you select a zip file, a list of the contained MIBs is displayed. You can select one or more of these MIBs to upload.

A list of MIB dependencies is displayed. A green check mark indicates that the dependency file has been found on the server.

Figure 6-4 MIB Dependencies



Add From MIB

Uploaded Mib: ACCOUNTING-CONTROL-MIB

All dependecies for the uploaded Mib were found. Override any exsiting dependencies if required and click Reanalyze.

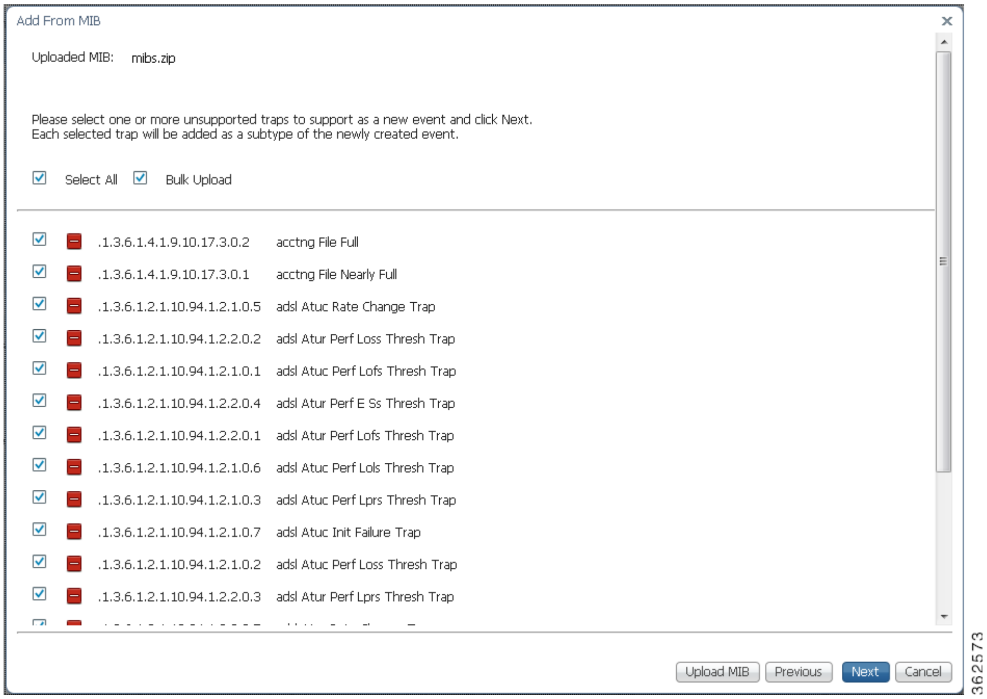
Click on Next in order to support traps from this Mib

☑ IANAifType-MIB
☑ IF-MIB
☑ SNMPv2-CONF
☑ SNMPv2-MIB
☑ SNMPv2-SMI
☑ SNMPv2-TC

Step 6 If any of the dependencies is not found, click **Browse** and select the dependency file to upload.

Step 7 Click **Next**. A list of traps is displayed. A red icon to the left of the trap indicates that it is not supported. A green icon indicates that the trap is already supported.

Figure 6-5 *Trap Selection*



Add From MIB

Uploaded MIB: mibs.zip

Please select one or more unsupported traps to support as a new event and click Next.
Each selected trap will be added as a subtype of the newly created event.

☑ Select All ☑ Bulk Upload

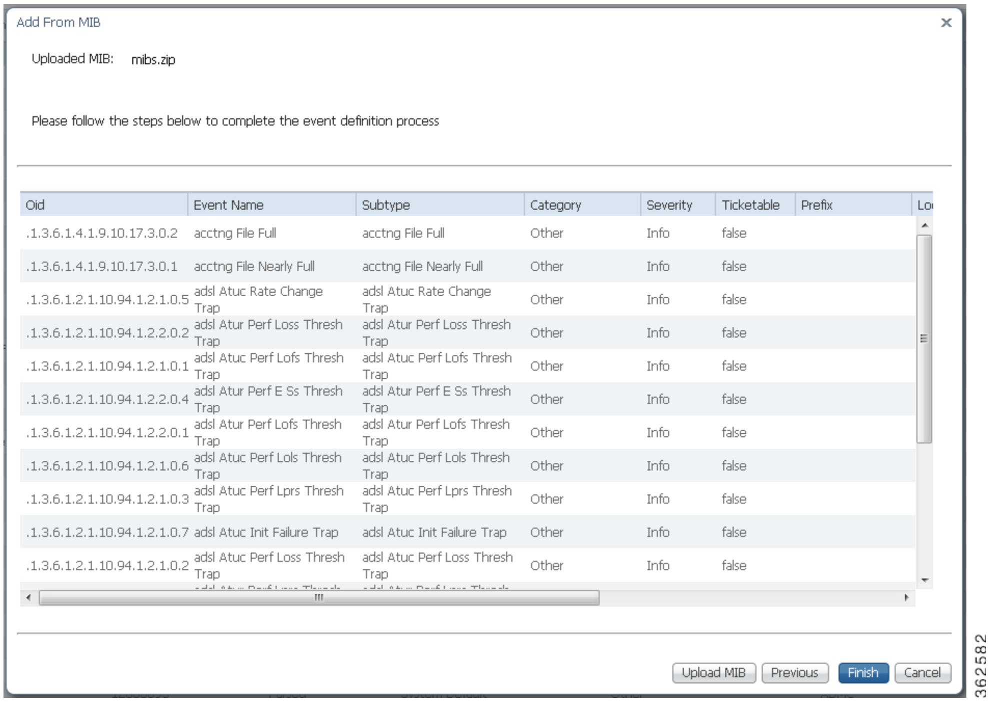
☑	.1.3.6.1.4.1.9.10.17.3.0.2	acctng File Full
☑	.1.3.6.1.4.1.9.10.17.3.0.1	acctng File Nearly Full
☑	.1.3.6.1.2.1.10.94.1.2.1.0.5	adsl Atuc Rate Change Trap
☑	.1.3.6.1.2.1.10.94.1.2.2.0.2	adsl Atur Perf Loss Thresh Trap
☑	.1.3.6.1.2.1.10.94.1.2.1.0.1	adsl Atuc Perf Lofs Thresh Trap
☑	.1.3.6.1.2.1.10.94.1.2.2.0.4	adsl Atur Perf E Ss Thresh Trap
☑	.1.3.6.1.2.1.10.94.1.2.2.0.1	adsl Atur Perf Lofs Thresh Trap
☑	.1.3.6.1.2.1.10.94.1.2.1.0.6	adsl Atuc Perf Lols Thresh Trap
☑	.1.3.6.1.2.1.10.94.1.2.1.0.3	adsl Atuc Perf Lprs Thresh Trap
☑	.1.3.6.1.2.1.10.94.1.2.1.0.7	adsl Atuc Init Failure Trap
☑	.1.3.6.1.2.1.10.94.1.2.1.0.2	adsl Atuc Perf Loss Thresh Trap
☑	.1.3.6.1.2.1.10.94.1.2.2.0.3	adsl Atur Perf Lprs Thresh Trap

Step 8 Select the Bulk Upload check box.

Step 9 Select the traps you want to add or check the Select All check box.

Step 10 Click **Next**. At this point, an event is created for each selected trap. The traps are listed in a table. Default values are provided for all mandatory attributes of the traps. You can edit these values inline in the table.

Figure 6-6 Trap Table



Uploaded MIB: mibs.zip

Please follow the steps below to complete the event definition process

Oid	Event Name	Subtype	Category	Severity	Ticketable	Prefix	Lo
.1.3.6.1.4.1.9.10.17.3.0.2	acctng File Full	acctng File Full	Other	Info	false		
.1.3.6.1.4.1.9.10.17.3.0.1	acctng File Nearly Full	acctng File Nearly Full	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.1.0.5	adsl Atuc Rate Change Trap	adsl Atuc Rate Change Trap	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.2.0.2	adsl Atur Perf Loss Thresh Trap	adsl Atur Perf Loss Thresh Trap	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.1.0.1	adsl Atuc Perf Lofs Thresh Trap	adsl Atuc Perf Lofs Thresh Trap	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.2.0.4	adsl Atur Perf E Ss Thresh Trap	adsl Atur Perf E Ss Thresh Trap	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.2.0.1	adsl Atur Perf Lofs Thresh Trap	adsl Atur Perf Lofs Thresh Trap	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.1.0.6	adsl Atuc Perf Lols Thresh Trap	adsl Atuc Perf Lols Thresh Trap	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.1.0.3	adsl Atuc Perf Lprs Thresh Trap	adsl Atuc Perf Lprs Thresh Trap	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.1.0.7	adsl Atuc Init Failure Trap	adsl Atuc Init Failure Trap	Other	Info	false		
.1.3.6.1.2.1.10.94.1.2.1.0.2	adsl Atuc Perf Loss Thresh Trap	adsl Atuc Perf Loss Thresh Trap	Other	Info	false		

Step 11 Optional. You can combine multiple traps in a single event as subtypes of the event. To do this, make the event name the same for all the traps you want to combine in a single event.

Step 12 Click **Finish**. A dialog box displays a list of the traps that were added.

Step 13 If you want to enable support for additional traps, click **Select Different Trap**, otherwise click **Close**. The traps you added now appear in the table of supported traps. You can edit a trap at any stage by selecting it and clicking **Edit**.

Enabling Support for a Custom Syslog

In this example, a custom syslog is generated by a router, using Embedded Event Manager (EEM), when the Windows XP server being monitored is not reachable. The custom syslog is %HA_EM-6-LOG: IPSLA-XP: Windows-XP unreachable. This event is sent to Prime Network but is not recognized or parsed.

Use the VCB to add support for this custom syslog, as follows:

Step 1 In the VCB tool, go to the VNE Drivers tab.

Step 2 Click on the arrow next to the VNE driver that represents the router that generates the custom syslog to expand its display. The Syslog Parsing Rule field shows the parsing rule used to parse events for this VNE driver, for both Product and IpCore schemes.

Step 3 Click the Syslog Parsing Rule link to show a list of syslog events associated with this parsing rule.

Step 4 Click the Add Row button to start defining the new syslog.

Step 5 Enter a unique name for the syslog in the Event Name field. For example, Monitoring XP Server.

Step 6 Check the Flapping check box if you want to define the criteria by which this trap will be considered a flapping event. This is relevant for network or communication related events that potentially could change state frequently. For example, link up/down, BGP neighbor up/down. When an event is flapping, Prime Network generates a single event indicating the flapping status and does not generate an additional event for each state change.
Enter the following information to define flapping criteria:

- Clear Interval—The flapping event will be cleared when the flapping has stopped for the specified time interval (in milliseconds).
- Flapping Interval and Flapping Threshold—To be defined as a flapping event, the event must change state every X milliseconds (Flapping Interval), and must occur Y consecutive times (Flapping Threshold). For example, if the Flapping Interval is 60000 msecs (one minute) and the Flapping Threshold is 5, the event must change state five times, every one minute, in order to be considered flapping.
- Update Interval and Update Threshold—An update event indicating that the event is still flapping will be sent either when the specified Update Interval has passed or when the event is received more times than the specified Update Threshold. For example, if the Update Interval is 100000 msecs and the Update Threshold is 500, an update event will be generated if the event remains flapping for longer than 100000 msecs or if the event is received more than 500 times, whichever occurs first.

Step 7 Click **Next** to go to the next step in the wizard which is to define the event subtypes. If you selected the Flapping check box in the previous step, you will see that flapping subtypes have been created. By default, the flapping subtype consists of the event name and the flapping subtype name. This can be changed as required. You can also change the severity of the flapping subtype so that it matches the severity of the event, for example.

Step 8 Enter the following information to define the first event subtype:

Field	Description
Name	Enter a unique name for the event subtype, for example, XP server inaccessible.
Description	Enter a string that describes the event, for example, “The XP server cannot be reached.”
Severity	Select the severity to be attributed to the event.
Ticketable	Check the check box if you want Prime Network to create a ticket for this event if there is no root cause event to which it can be correlated.
Auto Clear	Check the check box if you want Prime Network to automatically clear the event, without waiting for a clear event or for manual clearing of the event. If the auto clear check box is checked, the event will be cleared automatically 4 minutes after the last modification.
Correlate	Check the check box if you want the event to be correlated to a root-cause alarm.

Step 9 Click **Add** to define a second subtype, for example, XP server accessible, with severity “Cleared”.

Step 10 Click **Next** to go to the next step in the wizard which is identification and association of the event. In this step, you will provide an example of the raw event and you will define parameters by which the event will be identified.

Step 11 Enter the following information to define event identification and association.

Note You can provide multi-word values for the following parameters in the Event Identification and Association section while creating a syslog event: Interface Identifier, Interface Name, Substring(s) to Ignore, Subtype Key, and Instance Identifier Prefix.

<table>
<tr><th>Field</th><th colspan="2">Description</th></tr>
<tr><td>Raw Event</td><td colspan="2">Provide the raw event syntax as an example, so that the system can parse it.</td></tr>
<tr><td>Subtype Key</td><td colspan="2">Keyword that identifies the subtype. The keyword should be taken from the raw event. In this example, the key would be “unreachable”.</td></tr>
<tr><td rowspan="5">Source Type</td><td colspan="2">Select the source component of the device from which the event is generated. For this example, select ManagedElement Key because there is no specific interface or other component from which the event is generated. In other cases, you might choose from the following options:</td></tr>
<tr><td>Efp Key From Ifname Serviceid</td><td>Associates the event with a specific EFP DC, based on the EFP ID and the interface name.</td></tr>
<tr><td>Interface Key From Ifname</td><td>Associates the event with a specific interface that you specify in the Interface Name field.</td></tr>
<tr><td>Logical Container Key</td><td>Associates the event with a designated logical container that you select in the Container Type field.</td></tr>
<tr><td>ModuleDC With SlotSubslot Value Key</td><td>Associates the event with the corresponding module, based on the slot number.</td></tr>
<tr><td>Instance Identifier Prefix</td><td colspan="2">The unique identifier prefix that you specify in the new event location string which helps you to differentiate between the same event on the same interface but with a different CLI. The instance identifier prefix you provide will be displayed in Cisco Prime Network Vision and Cisco Prime Network Events along with the location identifier.</td></tr>
<tr><td>Interface Identifier</td><td colspan="2">Specify a value by which the interface will be identified (ifIndex is used to identify the interface).</td></tr>
<tr><td>Instance Identifier</td><td colspan="2">Specify the identifier of the instance. For this example, the instance identifier could be Windows XP.</td></tr>
<tr><td>Substring(s) to Ignore</td><td colspan="2">Prime Network filters the events that need to be parsed and processed based on the value(s) specified in this argument.</td></tr>
</table>

Step 12 Click **Next** to go to the next step in the wizard which is identification of the event subtypes. In this step, you will define values for each of the subtypes. In this example, the values could be “unreachable” for XP server inaccessible and “reachable” for XP server accessible.

Step 13 Optionally, go to **Step 5 - Expedite** if you want the device to be polled immediately for inventory updates upon receipt of the event, instead of waiting for the next polling cycle which may take up to twenty four hours for VNEs using reduced polling. Click **Add** to create an expedite rule and provide the following information:

Field	Description
Level	• Select Component if the expedite command will run on a component of the VNE. • Select Technology if the expedite command will run on the VNE level.
Class	Shows the VNE-level or the component-level command classes. Select the command class that contains the relevant expedite command.
Registration Name	Shows the commands belonging to the selected class. Select the relevant command.

Field	Description
Delay	Delay (in milliseconds). By default, the command will be executed immediately upon receipt of the event (default value is 0).
Key	Component level only. This is the entity by which the component will be identified. If the source type is the same as the source type specified for association, there is no need to specify the key again here.

Note Command expedition might result in unnecessary polling of devices which puts a load on the device and might affect performance. Therefore, before creating command expedition rules, consider whether the event is specific to a particular VNE or common to all VNEs and might affect the performance of VNEs using regular polling.

Step 14 Click **Next** to go to the next step in the wizard which determines which VNE drivers will be extended to support this event. This is determined by selecting the parsing rule groups per scheme that will be extended. The event will be supported on all VNE drivers that use the specified parsing rule groups.

Step 15 Click **Add** to select additional parsing rule groups, as required. In this example, all VNE drivers associated with group cisco-syslog-product-parsing-rules will be extended to support the new syslog. You can select additional groups for the Product scheme or you can select the IpCore scheme and a parsing rule group.

Note Certain parsing rules groups inherit from other groups. If you select multiple groups, make sure that your selection does not include a base (parent) group as well as the group that inherits from the base group. See Parsing Rule Group Inheritance Structure, page 6-13 for the relationship between parsing rule groups.

Step 16 Click **Finish**. The event now appears in the list of syslogs for the cisco-syslog-product-parsing-rules group.

Parsing Rule Group Inheritance Structure

Table 6-1 shows the inheritance structure for parsing rule groups. The groups on the left inherit settings from the groups on the right. When you select a group that is high in the hierarchy, all of the groups (and subgroups) that inherit from it are included. You should not choose the lower groups to make sure they are included. For example:

cisco-**asr90xx-syslog-ipcore**-parsing-rules

inherits from:

cisco-**iox-syslog-ipcore**-parsing-rules

which inherits from:

cisco-**iox-syslog-product**-parsing-rules

If you choose cisco-**iox-syslog-product**-parsing-rules (which is highest in the hierarchy), do not choose the other two groups; they are automatically included.

Table 6-1 Parsing Rule Groups Inheritance

This Parsing Rule Group...	Inherits Settings from This Group:
cisco-asr90xx-syslog-ipcore-parsing-rules	cisco-iox-syslog-ipcore-parsing-rules
cisco-asr90xx-trap-ipcore-parsing-rules	cisco-iox-trap-ipcore-parsing-rules
cisco-ciscocpt-trap-ipcore-parsing-rules	cisco-trap-ipcore-parsing-rules
cisco-ciscocpt-trap-product-parsing-rules	cisco-trap-product-parsing-rules
cisco-ciscocpt-syslog-ipcore-parsing-rules	cisco-syslog-ipcore-parsing-rules
cisco-ciscocpt-syslog-product-parsing-rules	cisco-syslog-product-parsing-rules
cisco-iox-syslog-ipcore-parsing-rules	cisco-iox-syslog-product-parsing-rules
cisco-me26xx-syslog-ipcore-parsing-rules	cisco-syslog-ipcore-parsing-rules
cisco-me26xx-syslog-product-parsing-rules	cisco-syslog-product-parsing-rules
cisco-rfgw-trap-product-parsing-rules	cisco-trap-product-parsing-rules
cisco-syslog-ipcore-parsing-rules	cisco-syslog-product-parsing-rules
cisco-trap-ipcore-parsing-rules	cisco-trap-product-parsing-rules
nexus-trap-product-parsing-rules	nexus-422v-trap-product-parsing-rules

Testing Your New Events

After you have enabled support for a trap or syslog in the VCB, you can simulate the event to check that the operation has been successful. After you simulate the event and send it to the Prime Network gateway, it should appear in the Prime Network Events and Prime Network Vision GUI.

To test an event:

Step 1 Select the event you want to test.

Step 2 Click **Test Event**.

Step 3 In the displayed Test Event dialog, select the device on which you want to simulate the event in the Device IP field.

Step 4 Provide any required information. For syslogs, the message box is automatically populated only if the message was included in the parsing rules when the event was created.

Step 5 Click **Send**. The event will be sent to the specified device.

Step 6 Open Prime Network Vision or Prime Network Events and check that the event appears in the list of events for the device. If it is not a correlated event, it will appear in the GUI almost immediately after the event is sent. If it is a correlated event, it could take up to three minutes before it appears in the GUI.

Step 7 If the event does not appear in the GUI, you can troubleshoot by following the instructions in Troubleshooting Event Customization Using the VCB CLI, page B-53.

Customizing Events

Using the VCB, you can change the way Prime Network deals with events. For example, you can change the severity of an event, or you can instruct the system to drop the event. Event customization is described in the following sections:

- Changing the Severity of an Event Subtype, page 6-15
- Dropping an Event, page 6-15
- Restoring a System Default Event, page 6-16
- Deleting Events, page 6-16
- Displaying MIB Names Instead of Numbers in Trap Details, page 6-17
- Adding Recommended Actions to Event Details, page 6-17
- Adding Parameter Values to Event Descriptions, page 6-19

Changing the Severity of an Event Subtype

The events that are supported by default in Prime Network are attributed with a specific severity. You can customize the event and change the severity if it is not appropriate for your organization. For example, the Prime Network system considers the event, “ASR5 port down” to have a severity of “Warning”. However, in your organization, this event might be considered to be “Major” and you want the event to be marked as such.

To change the severity of an event subtype:

Step 1 In the VCB tool, select **Tools > Events** or click on the Events tab.

Step 2 Click the arrow next to the event you want to customize to expand its display.

Step 3 Select the required severity from the Severity drop-down menu and click **Save**.

Dropping an Event

By default, when an event is received by Prime Network, it is archived and parsed. Only events that have been parsed will appear in Prime Network Events tables. You can choose to drop an event so that it no longer appears in the tables. The event will no longer be actionable, meaning that it will not be processed and parsed, but it will be archived. In the case of service events, the event will no longer be generated by the system so there will be no archiving.

When an event has been dropped, it cannot longer be edited or updated (but it can be restored).

To drop an event:

Step 1 In the VCB tool, select **Tools > Events** or click on the Events tab.

Step 2 Select the event you want to drop and click **Modify Inbound Handling**.

Step 3 Click **OK** in the confirmation message. The Inbound Handling column for the event will change to Archived Only for syslogs and traps or to Disabled for system events.

Restoring a Dropped Event

To restore a dropped event:

Step 1 In the VCB tool, select **Tools > Events** or click on the Events tab.

Step 2 Select the event you want to restore and click **Modify Inbound Handling**.

Step 3 Click **OK** in the confirmation message. The Inbound Handling column for the event will change to Parsed.

Restoring a System Default Event

If you have edited a system default event and you want to go back to the original event, you can restore the system default event.

The Overriding System Default column indicates whether or not a system default event has been edited. The values for this column are true or false.

Note A VNE upgrade package might provide support for events that you previously added using the VCB. After you have upgraded the VNE driver, such events are marked as overriding the system default. Use this procedure to restore the system default event that is provided with the upgrade.

To restore a system default event:

Step 1 In the VCB tool, select **Tools > Events** or click on the Events tab.

Step 2 Select the event you want to restore and click **Restore**.

Step 3 Click **OK** in the confirmation message. The Overriding System Default column for the event will change to False.

Deleting Events

User-defined events can be deleted as long as they are not overrides of system default events. System default events cannot be deleted.

Note For system default overrides, a Restore button is provided instead of the Delete button.

To delete an event:

Step 1 In the VCB tool, select **Tools > Events** or click on the Events tab.

Step 2 Select the event you want to delete and click **Delete**.

Step 3 Click **OK** in the confirmation message. The event is removed from the table.

Displaying MIB Names Instead of Numbers in Trap Details

Prime Network has a repository of pre-compiled MIBs that are used to resolve trap OIDs into a more readable and user-friendly form. When you use the VCB to add support for new traps via MIB, the repository is automatically updated and the trap details in Vision or Events contains the MIB name rather than the dotted notation representation of the MIB.

For supported traps that are showing unresolved trap OIDs in the trap details, you can manually compile the MIB and update the repository. After doing this, the trap details will show the MIB name instead of numbers. To compile the MIB in order to display the MIB name in the trap details:

Step 1 In the VCB tool, select **Tools > Events** or click on the Events tab.

Step 2 Click on the Trap tab.

Step 3 Click **Add from MIB**.

Step 4 Upload the relevant MIB repository file or zip file and click **Next**.

Step 5 Select the MIB from which the trap is generated and click **Next**.

Step 6 Select the Compile Only check box.

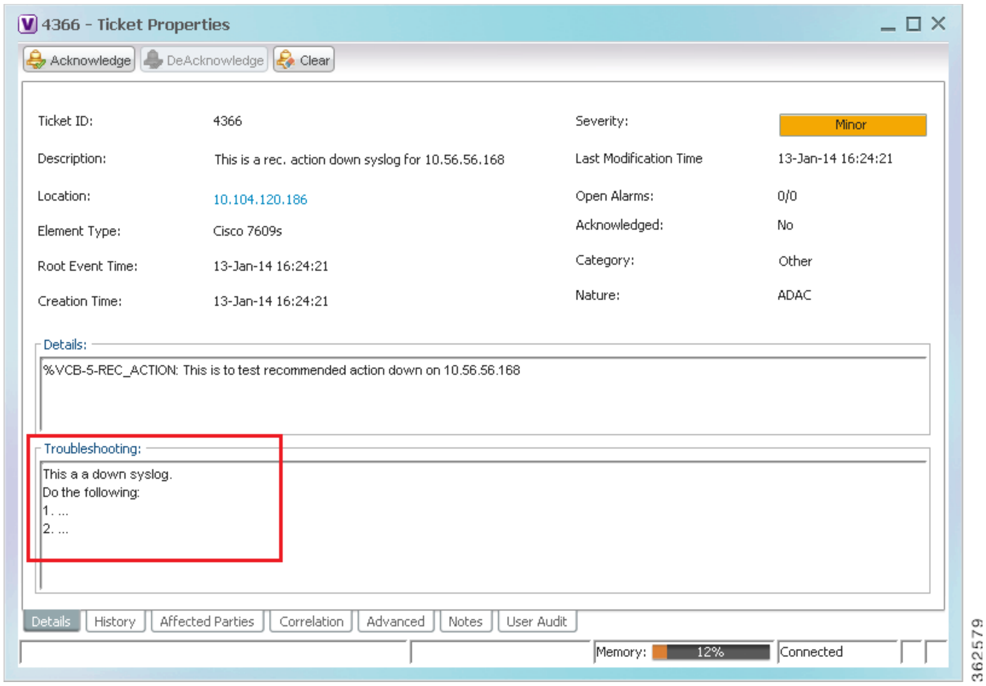
Step 7 Click **Finish**.

Step 8 Verify that the trap OID has been translated by simulating the trap and viewing the trap details in the Prime Network Vision GUI. See Testing Your New Events, page 6-14, for instructions on how to simulate events.

Adding Recommended Actions to Event Details

Using the VCB, you can enhance the event details and make them more useful by outlining the actions that should be performed after receiving the event. The recommended actions will appear in the Troubleshooting section of the event or ticket properties (Details tab).

Figure 6-7 Example of Customized Recommended Actions for an Event



Ticket ID: 4366
Description: This is a rec. action down syslog for 10.56.56.168
Location: 10.104.120.186
Element Type: Cisco 7609s
Root Event Time: 13-Jan-14 16:24:21
Creation Time: 13-Jan-14 16:24:21
Severity: Minor
Last Modification Time 13-Jan-14 16:24:21
Open Alarms: 0/0
Acknowledged: No
Category: Other
Nature: ADAC

Details:
%VCB-5-REC_ACTION: This is to test recommended action down on 10.56.56.168

Troubleshooting:
This a a down syslog.
Do the following:
1. ...
2. ...

Recommended actions can be added when adding new traps or syslogs or at a later stage, by editing existing events.

For some traps, troubleshooting information is automatically extracted from the MIB and populated in the ticket/event properties. You can edit this information using the VCB. This information is not available for syslogs, therefore you would have to add recommended actions using the VCB, as described below.

To add recommended actions to existing traps or syslogs:

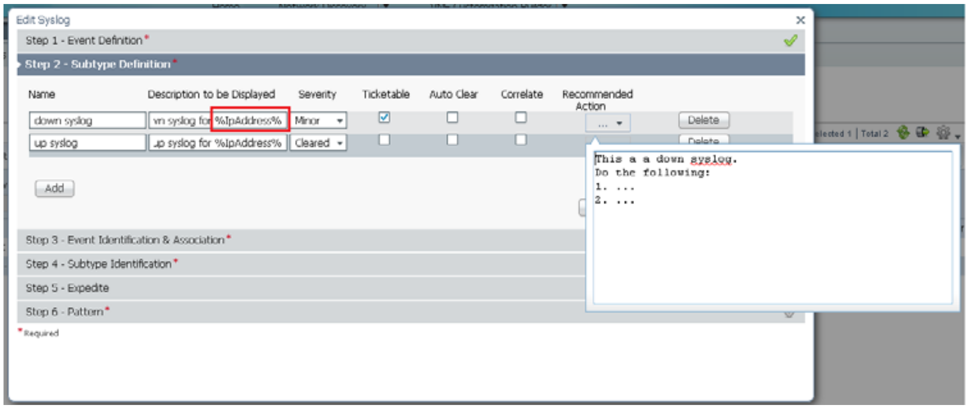
Step 1 In the VCB tool, select **Tools > Events** or click on the Events tab.

Step 2 Select the event for which you want to add recommended actions and expand its display in the table by clicking on the arrow on the left.

Step 3 Click on the arrow in the Recommended Action column to display a text box which allows you to enter free text.

Note If you are adding a new event or editing an event, this column is located in the Subtype Definition step of the wizard.

Figure 6-8 Adding the Recommended Actions


Edit Syslog
Step 1 - Event Definition *
Step 2 - Subtype Definition *
Add
Delete
Step 3 - Event Identification & Association *
Step 4 - Subtype Identification *
Step 5 - Expedite
Step 6 - Pattern *
* Required
362580

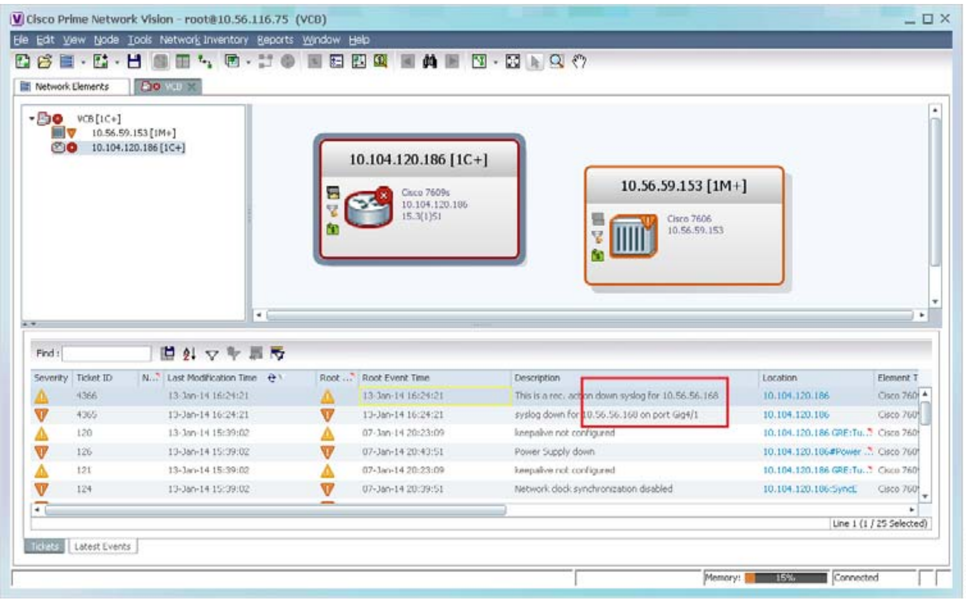
Name	Description to be Displayed	Severity	Ticketable	Auto Clear	Correlate	Recommended Action
down syslog	vn syslog for %IpAddress%	Minor	☑	□	□	...
up syslog	up syslog for %IpAddress%	Cleared	□	□	□	

```
This a a down syslog.
Do the following:
1. ...
2. ...
```

Step 4 Enter the recommended actions in the text box and click **Save**.

Step 5 Verify that the recommended actions have been added successfully by simulating the event and viewing the event details in the Prime Network Vision GUI. (See Testing Your New Events, page 6-14, for instructions on how to simulate events.)

Figure 6-9 Simulating the Event with Customized Recommended Actions



Severity	Ticket ID	N...	Last Modification Time	Root ...	Root Event Time	Description	Location	Element T
⚠	4366		13-Jan-14 16:24:21	⚠	13-Jan-14 16:24:21	This is a rec. action down syslog for 10.56.56.168	10.104.120.186	Cisco 760
▼	4365		13-Jan-14 16:24:21	▼	13-Jan-14 16:24:21	syslog down for 10.56.56.168 on port Gig4/1	10.104.120.186	Cisco 760
⚠	120		13-Jan-14 15:39:02	⚠	07-Jan-14 20:23:09	keepalive not configured	10.104.120.186 GRE:Tu...	Cisco 760
▼	126		13-Jan-14 15:39:02	▼	07-Jan-14 20:43:51	Power Supply down	10.104.120.186#Power ...	Cisco 760
⚠	121		13-Jan-14 15:39:02	⚠	07-Jan-14 20:23:09	keepalive not configured	10.104.120.186 GRE:Tu...	Cisco 760
▼	124		13-Jan-14 15:39:02	▼	07-Jan-14 20:39:51	Network clock synchronization disabled	10.104.120.186:SyncE	Cisco 760

Adding Parameter Values to Event Descriptions

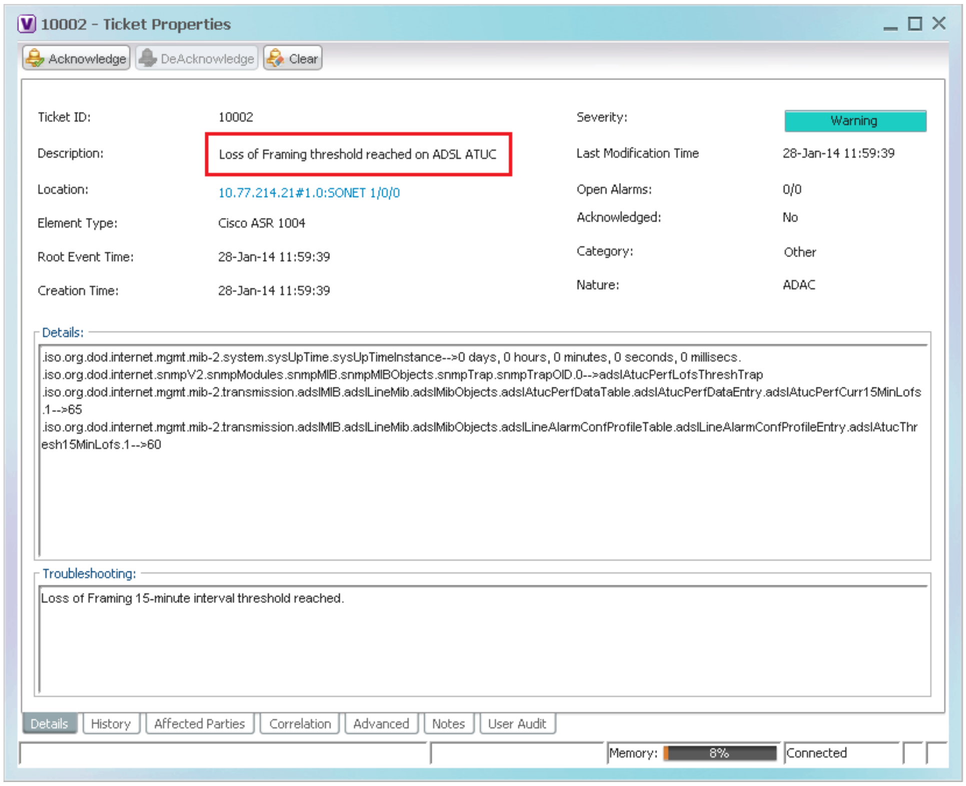
In previous versions of Prime Network, the event description was a static string and could not be changed. From Prime Network 4.2.2, you can define more descriptive event descriptions by including runtime parameters with dynamic values, such as interface names, threshold levels, and so on. These parameter values will be included in the event description displayed in Prime Network Vision or Events, making it easier to identify, at a glance, what the event is about without having to drill down to the event details.

This functionality applies to user-defined events only (not system-defined events). You can add parameter values to event descriptions when adding a new trap/syslog or when editing a trap/syslog.

Note When doing bulk upload of traps, parameter values cannot be added to event descriptions.

See Figure 6-10 and Figure 6-11 below for an example of a less detailed and more detailed event description.

Figure 6-10 Event Description Before Adding Dynamic Runtime Parameter Values



Ticket ID:	10002	Severity:	Warning
Description:	Loss of Framing threshold reached on ADSL ATUC	Last Modification Time	28-Jan-14 11:59:39
Location:	10.77.214.21#1.0:SONET 1/0/0	Open Alarms:	0/0
Element Type:	Cisco ASR 1004	Acknowledged:	No
Root Event Time:	28-Jan-14 11:59:39	Category:	Other
Creation Time:	28-Jan-14 11:59:39	Nature:	ADAC

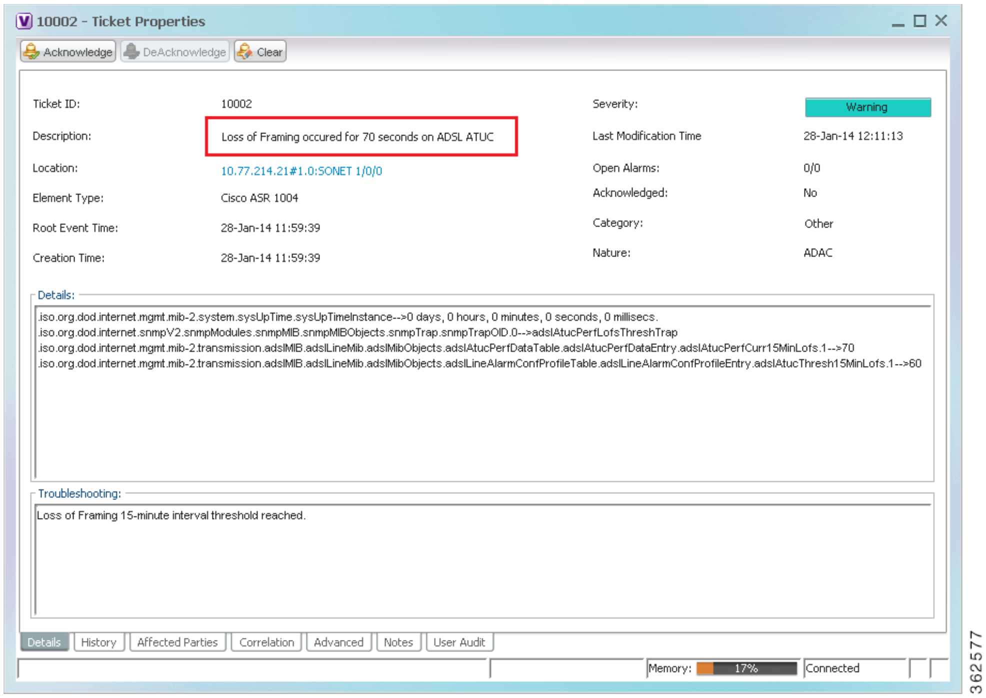
Details:

```
.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.sysUpTimeInstance-->0 days, 0 hours, 0 minutes, 0 seconds, 0 millisecs.
.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObjects.snmpTrap.snmpTrapOID.0-->adslAtucPerfLofsThreshTrap
.iso.org.dod.internet.mgmt.mib-2.transmission.adslMIB.adslLineMib.adslMibObjects.adslAtucPerfDataTable.adslAtucPerfDataEntry.adslAtucPerfCurr15MinLofs
.1-->65
.iso.org.dod.internet.mgmt.mib-2.transmission.adslMIB.adslLineMib.adslMibObjects.adslLineAlarmConfProfileTable.adslLineAlarmConfProfileEntry.adslAtucThr
esh15MinLofs.1-->60
```

Troubleshooting:

Loss of Framing 15-minute interval threshold reached.

Figure 6-11 Event Description with New Dynamic Runtime Parameter Values



Ticket ID:	10002	Severity:	Warning
Description:	Loss of Framing occured for 70 seconds on ADSL ATUC	Last Modification Time	28-Jan-14 12:11:13
Location:	10.77.214.21#1.0:SONET 1/0/0	Open Alarms:	0/0
Element Type:	Cisco ASR 1004	Acknowledged:	No
Root Event Time:	28-Jan-14 11:59:39	Category:	Other
Creation Time:	28-Jan-14 11:59:39	Nature:	ADAC

Details:

```
.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.sysUpTimeInstance-->0 days, 0 hours, 0 minutes, 0 seconds, 0 millisecs.
.iso.org.dod.internet.snmpV2.snmpModules.snmpMIB.snmpMIBObjects.snmpTrap.snmpTrapOID.0-->adslAtucPerfLofsThreshTrap
.iso.org.dod.internet.mgmt.mib-2.transmission.adslMIB.adslLineMib.adslMibObjects.adslAtucPerfDataTable.adslAtucPerfDataEntry.adslAtucPerfCurr15MinLofs.1-->70
.iso.org.dod.internet.mgmt.mib-2.transmission.adslMIB.adslLineMib.adslMibObjects.adslLineAlarmConfProfileTable.adslLineAlarmConfProfileEntry.adslAtucThresh15MinLofs.1-->60
```

Troubleshooting:

Loss of Framing 15-minute interval threshold reached.

To add parameter values to event descriptions:

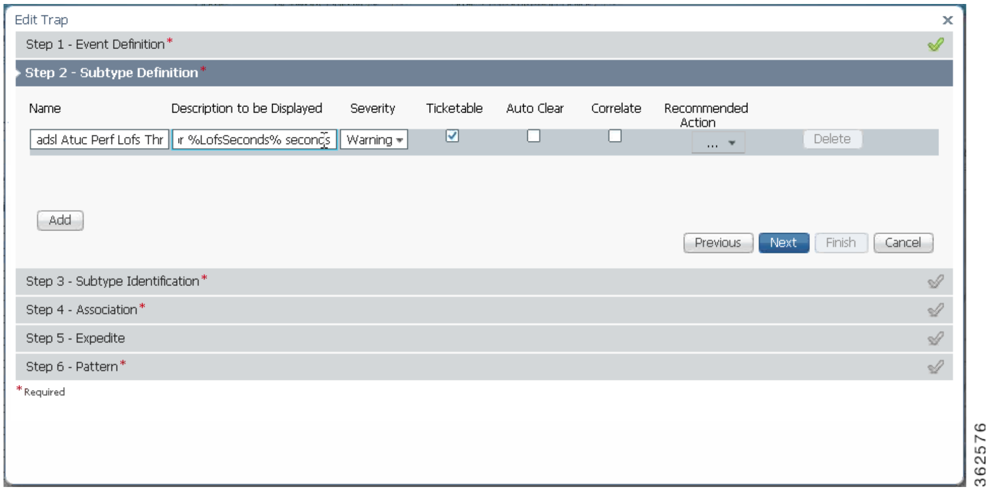
Step 1 In the VCB tool, select **Tools > Events** or click on the Events tab.

Step 2 Select the required trap/syslog and click **Edit**.

Step 3 Go to **Subtype Definition**.

Step 4 In the Description to be Displayed field, use a percentage (%) delimiter and specify the required parameter, for example, **%LofsSeconds%**.

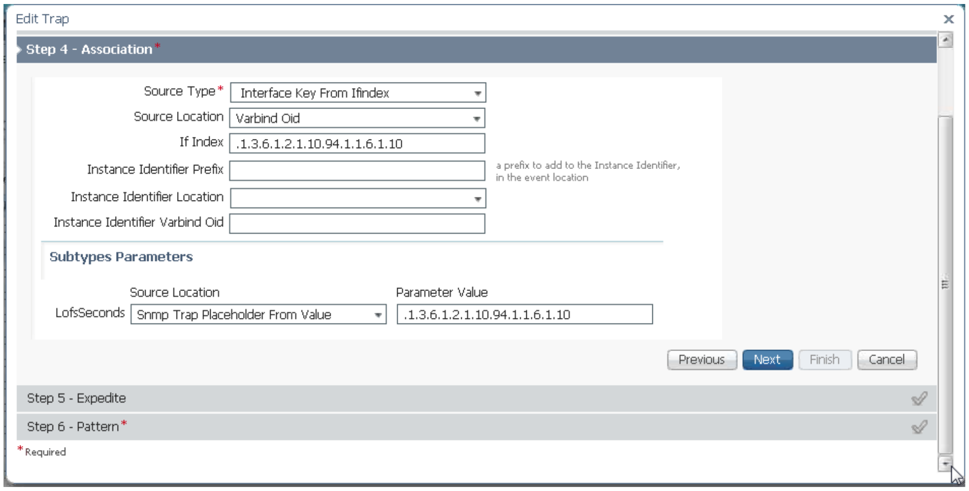
Figure 6-12 Subtype Definition - Add Parameter to Description



Name	Description to be Displayed	Severity	Ticketable	Auto Clear	Correlate	Recommended Action
adsl Atuc Perf Lofs Thr	ır %LofsSeconds% seconds	Warning	☑	□	□	...

Step 5 Go to **Association**. The parameter you included in the description will appear under Subtype Parameters.

Figure 6-13 Association



Edit Trap

Step 4 - Association*

Source Type* Interface Key From Ifindex

Source Location Varbind Oid

If Index .1.3.6.1.2.1.10.94.1.1.6.1.10

Instance Identifier Prefix

a prefix to add to the Instance Identifier, in the event location

Instance Identifier Location

Instance Identifier Varbind Oid

Subtypes Parameters

	Source Location	Parameter Value
LofsSeconds	Snmp Trap Placeholder From Value	.1.3.6.1.2.1.10.94.1.1.6.1.10

Step 6 Go to **Pattern** and click **Finish**.

Step 7 Verify that the parameter values have been added successfully by simulating the trap and viewing the event description in the Prime Network Vision GUI. See Testing Your New Events, page 6-14, for instructions on how to simulate events.

CHAPTER 7

Adding New Threshold-Crossing Alarms

Prime Network enables the creation of Threshold-Crossing Alarms (TCAs), which are generated when a specific condition is met. These alarms can be viewed in the Prime Network Vision and Events GUI clients.

TCAs are associated with soft properties. A soft property is a mechanism for extending the NE information that Prime Network models. The soft property contains rules for retrieving new NE information, parsing the retrieved information, and displaying it in the Inventory window. To create a TCA, you can apply it to an existing soft property, or create a new soft property and associate a TCA to it. For information on creating soft properties, see Steps for Adding New NE Property Information to the Inventory, page 2-3.

This topic explains how to create TCAs for soft properties:

- Creating a New Threshold-Crossing Alarm, page 7-1

Creating a New Threshold-Crossing Alarm

The procedure for creating a TCA is an extension of the procedure for creating a soft property, described in Displaying a New Property in the Inventory Window, page 2-4. You can create a TCA for an existing soft property, or create a new soft property and associate a TCA to it. You can create multiple TCAs for the same soft property. If necessary, the TCA can be disabled, in which case it will not be triggered for the specified condition until it is re-enabled.

TCAs are only supported on Property type soft properties.

Before You Begin

- Locate the soft property for which you want to define an alarm condition by right-clicking the relevant NE and choosing **Management > Soft Properties Management**.
- If you need to create a new soft property, follow the procedure in Displaying a New Property in the Inventory Window, page 2-4 or Adding a New Inventory Table Containing SNMP get Details, page 2-11

To create a TCA:

Step 1 In Prime Network Vision, right-click a relevant NE and choose **Management > Soft Properties Management**.

The Soft Properties Manager dialog box opens and displays any existing soft properties for the selected NE.

Step 2 If you are creating a new soft property, choose **File > New Element** and follow the procedure in Displaying a New Property in the Inventory Window, page 2-4.

or

If you are adding a TCA for an existing soft property, choose **File > Edit Element**. This displays a window listing the complete hierarchy for the soft property. Checks indicate the classes to which the soft property is available.

Step 3 In the Edit Soft Property dialog box, locate the local instance of the soft property; this is most likely at the very bottom of the table. Select it and click **OK**.

Step 4 Click the TCA Alarms tab. If the soft property has any defined TCAs, they will be listed.

Step 5 Click **Add** to open the Add TCA dialog box.

Step 6 Click the General tab and enter the basic alarm information.

Field	Description	Example
Name	Alarm name that is displayed in the ticket pane when the alarm is triggered.	**My value is not 5**
Enabled	Whether the VNE is monitoring for this alarm. This field allows you to temporarily disable a TCA, if needed.	(Enabled)
Description	(Optional) A description of the alarm.	**Show this alarm if the value is not equal to 5**
Alarm Severity	Choose the severity that is appropriate for this TCA in your network: Critical, Major, Minor, Warning, or Normal (clearing).	**Critical**
Can be correlated to other alarms	Enables correlating this alarm to other alarms.	(Not selected)
Other alarms can correlate to this alarm	Enables other alarms to correlate to this alarm.	(Not selected)

Step 7 Click the Trigger tab and define the conditions that will trigger the alarm.

Note For triggers that require numeric values, the parse integer rule must be applied on the soft property as an ending rule.

Trigger	Description
Value Equal	Generates an alarm when the when the soft property value is equal to the specified target value.
Value Not Equal	Generates an alarm when the when the soft property value is not equal to the specified target value.
Upper Threshold	Generates an alarm when the value exceeds the specified value. Clears this alarm when the value drops below the specified value. The upper threshold value must be numeric.
Lower Threshold	Generates an alarm when the value drops below the specified value. Clears this alarm when the value rises above the specified value. The lower threshold value must be numeric.
Upper Rate	Checks counter value changes over a period of one second. Generates an alarm when the specified upper rate is exceeded. Clears the alarm when the rate drops to the specified value. When this trigger is used with the *Trigger alarm only if change persists more than* option, you can check that the rate is maintained above the specified value over time. **Note** If the property is sampled every *x* seconds, the calculation is the *current value* less the *previous value* divided by *x* seconds.
Lower Rate	Checks counter value changes over a period of one second. Generates an alarm when the rate drops below the specified lower rate. Clears the alarm when the rate reaches the specified value. When this is used with the *Trigger alarm only if change persists more than* option, you can check that the rate is maintained below the specified value over time. **Note** If the property is sampled every *x* seconds, the calculation is the *current value* less the *previous value* divided by *x* seconds.

Step 8 To generate the alarm only if the trigger condition is maintained for a certain length of time, select the relevant check box and specify the length of time in seconds.

Step 9 Click **OK** to exit the Edit TCA dialog box.

Step 10 Click **OK** to exit the Edit Soft Property dialog box.

Step 11 Click **Close** to exit the Soft Properties Manager.

The TCA is activated on the NE. If the trigger conditions are met, you should see the event in the Tickets tab (Prime Network Vision) or tickets table (Prime Network Events). When you restart the VNE, the TCA is applied to all similar components on the VNE.

If you want to make the TCA available to other NEs, follow the procedure in Making the Soft Property Available to Other NEs, page 2-14.

PART 5

Extending the GUI Client

This topic explains how you can extend the Prime Network Vision GUI client by adding a launch point to an external application:

- Adding External Launch Points to the GUI Client, page 8-1

CHAPTER 8

Adding External Launch Points to the GUI Client

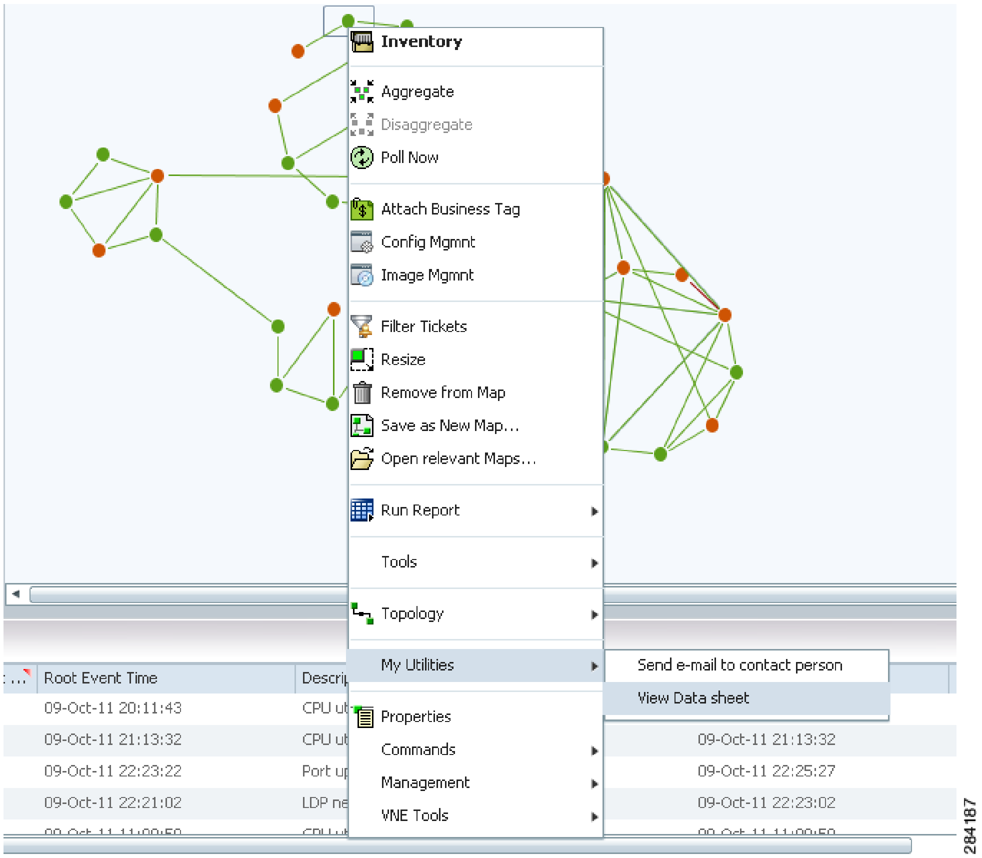
These topics describe how to add launch points from Prime Network to external applications:

- Overview: Adding External Launch Points, page 8-1
- Adding an External Launch Point to the GUI Client, page 8-3
- Deleting an External Launch Point, page 8-10
- Sample BQL Scripts for Launching External Applications, page 8-11

Overview: Adding External Launch Points

If you have configurator or higher privileges, Prime Network allows you to add right-click menu options to elements in order to launch external applications or URLs. These are called external launch points. When the external launch point is clicked, it invokes the external application (for example, a script or batch file), or opens the specified URL in the default browser. Figure 8-1 illustrates a new launch point named My Utilities.

Figure 8-1 External Launch Point Example



:...	Root Event Time	Descri	
	09-Oct-11 20:11:43	CPU ut	
	09-Oct-11 21:13:32	CPU ut	09-Oct-11 21:13:32
	09-Oct-11 22:23:22	Port up	09-Oct-11 22:25:27
	09-Oct-11 22:21:02	LDP ne	09-Oct-11 22:23:02
	09-Oct-11 11:09:50	CPU ut	09-Oct-11 11:09:50

You can add launch points to any IMO, including network elements, links, tickets, and events. You identify the IMO according to IMO type. The launch points appear as additional right-click menu options on all IMOs of the specified type.

You can narrow down the instances on which the external launch point appears by filtering by any of the properties of the IMO. You can also create a launch point on a property of a referenced IMO.

When you add a launch point, the site.xml file is updated on the Prime Network gateway. The new launch point (right-click menu option) will appear in all Prime Network clients that connect to the gateway.

The scripts or batch files used in the external launch point definition can be copied to one of the following locations:

- The Prime Network client machine—In this case, the launch point will only be functional on that client.
- The Prime Network gateway, under the Main/webstart/scripts directory—In this case, the scripts will be downloaded automatically to all clients associated with the gateway, upon next login. The launch points will be functional from all these clients. Please follow the instructions in Enabling Automatic Download of BQL Scripts to Clients, page 8-10, to enable the automatic download of the scripts to the clients.
- A shared location.

Adding an External Launch Point to the GUI Client

Adding an external launch point involves the following steps:

1. Identifying the IMO Context and Properties, page 8-3
2. Creating BQL Scripts, page 8-4
3. Running BQL Scripts, page 8-10
4. Enabling Automatic Download of BQL Scripts to Clients, page 8-10

Note You should maintain a list of all the IMO contexts where launch points have been added, as there is no way to get such a list from the system. This will enable you to update or delete external launch points at a later stage.

Identifying the IMO Context and Properties

To identify the IMO context and properties:

Step 1 Select a network element from the Tree pane.

Step 2 Press **F2**. The All Properties viewer appears in a separate window.

Step 3 Make a note of the IMO type. If you are in the Map pane, you should make a note of the IMO type that is part of ContainedIMO and not part of IHierarchyNode; for example, IPortConnector or IManagedElement.

Example 1: Identifying IManagedElement in the Tree pane:

```
com.sheer.imo.IManagedElement
{[ManagedElement(Key=ana7609-1)]}
```

Example 2: Identifying IPortConnector in the Tree pane:

```
com.sheer.imo.IPortConnector
{[ManagedElement(Key=ana7609-1)][PhysicalRoot][Chassis][Slot(SlotNum=5)][Module][Port(Port
Number=GigabitEthernet5/1 - Missing Pluggable Port)]}
```

Example 3: Identifying IManagedElement in the Map pane:

```
{[HierarchyNode(Id=1001)][ContainedImo]}
     key=Imo, value=com.sheer.imo.IManagedElement
        {[ManagedElement(Key=ana7609-1)]}
```

Step 4 Make a note of the property key name; for example, in IManagedElement write down IP; in IPortConnector write down PortAlias.

Example 1: IManagedElement

```
key=DeviceName, value=ana7609-1
key=IP, value=172.20.68.72
```

Example 2: IPortConnector

```
key=Location, value=5.GigabitEthernet5/1 - Missing Pluggable Port
key=PortAlias, value=GigabitEthernet5/1 - Missing Pluggable Port
```

Creating BQL Scripts

Read the *Cisco Prime Network Integration Developer Guide* for an understanding of Broadband Query Language (BQL), and as a prerequisite to understanding this section.

Table 8-1 describes the format and syntax that you must follow when writing the BQL **set** command to launch an external application:

Table 8-1 BQL Command Format and Syntax

Property / Type	Value	Description
management.IExternalLaunch		

`ID / OID`	Examples:
	For network elements:
	`{[ExternalLaunch(ContextImoType=com.sheer.imo.IManagedElement)(Name=example1)]}`
	For tickets:
	`{[ExternalLaunch(ContextImoType=com.sheer.imo.newalarm.ITicket)(Name=example36)]}`
		The OID contains the context and unique name of the external launch point.
		It defines the IMO type on which the external launch point is defined, for example:
		• IManagedElement—Network elements.
		• IEvent (or any other subtype, like ISyslogAlarm)—Events and their subtypes.
		• ITicket—Tickets.
		• ITopologicalLink—Links.

Table 8-1 BQL Command Format and Syntax (continued)

Property / Type	Value	Description
FilterParameters / String	Example: <Filter type="management.IExternalLaunchFilter" > <FilterParameters type="IMObjects_Array"> <management.IExternalLaunchFilterParameter type="management.IExternalLaunchFilterParameter" > <Value type="String">reg-exp</Value> <Property type="String">SysLocation</Property> </management.IExternalLaunchFilterParameter> <management.IExternalLaunchFilterParameter type="management.IExternalLaunchFilterParameter"> <Value type="String">reg-exp</Value> <Property type="String">ObjectId.Key</Property> </management.IExternalLaunchFilterParameter> <management.IExternalLaunchFilterParameter type="management.IExternalLaunchFilterParameter"> <Value type="String">reg-exp</Value> <Property type="String">@IBusinessObject.Notes</Property> </management.IExternalLaunchFilterParameter> <management.IExternalLaunchFilterParameter type="management.IExternalLaunchFilterParameter"> <Value type="String">reg-exp</Value> <Property type="String">~time~</Property> </management.IExternalLaunchFilterParameter> <management.IExternalLaunchFilterParameter type="management.IExternalLaunchFilterParameter"> <Value type="String">reg-exp</Value> <Property type="String">ScriptMetadataOids</Property> <Match type="String">all</Match> </management.IExternalLaunchFilterParameter> <management.IExternalLaunchFilterParameter type="management.IExternalLaunchFilterParameter"> <Value type="String">1-100</Value> <Property type="String">CpuUsage</Property> </management.IExternalLaunchFilterParameter> </FilterParameters> <Match type="String">any</Match> </Filter>	Enables you to apply a filter to narrow down the instances on which the external launch point will be available. For example, you can use this filter if you want only IMOs with specific properties to have the external launch point, rather than all IMOs of the specified type. You can use any of the IMO's properties in the filter. The filter property value can be: • Numbers—a pipe-delimited list of numbers and ranges, such as: "1|4-6|8" • Strings and other scalar properties—a Java regular expression (reg-exp) that is checked for containment, i.e., a full match is not required. Scalar properties are converted to string values, and null values are converted to empty strings (""). • Enum properties—either the enum number or string value. • Arrays—value is one of the above, with the option to specify whether any (at least one) or all of the items must match the filter value. Multiple filter criteria are also supported. The match any or match all option should include the logical condition between the criteria elements.

Table 8-1 BQL Command Format and Syntax (continued)

Property / Type	Value	Description

`ExternalBatchToExecute / String`	Examples:
	• Specifying a batch file located on the client machine:
	`<ExternalBatchToExecute type="String">C:\runthis.bat</ExternalBatchToExecute>`
	• Specifying a batch file located on the gateway machine, this batch file will be automatically downloaded from the gateway machine:
	`<ExternalBatchToExecute type="String">foldername/runthis.bat</ExternalBatchToExecute>`
	• Specifying a URL:
	`<ExternalBatchToExecute type="String">http:\\www.cisco.com</ExternalBatchToExecute>`
	• Passing an environment variable:
	`<ExternalBatchToExecute type="String">%java_home%\bin\javadoc</ExternalBatchToExecute>`
		Defines the details of the executable steps or batch file to be invoked. You:
		• Must specify the parameters as an array when using a script.
		• Cannot pass any parameters in the URL. You must use batch files or LineToExecute to pass the parameters.
		• Can include more commands as part of a batch file.
		• Can include an environment variable when invoking the batch file.
		To define a string that represents the URL or script which is executed along with its parameters, use the LinetoExecute property.

Table 8-1 BQL Command Format and Syntax (continued)

Property / Type	Value	Description

`LineToExecute / String`	Example:
	Launching a URL with the Gateway IP, network element IP, and logged-in username parameters, using / as delimiter:
	`<LineToExecute type="String">http://url.example.com/$GW$/$com.sheer.imo.IManagedElement.IP$/$USERNAME$</LineToExecute>`
	Example:
	Launching a URL from the soft property from the interface of the supported IMO:
	`<LineToExecute type="String">http://url.example.com/$GW$/$com.sheer.imo.IEthernetService.NetworkElemnt#softproperty$/$USERNAME$</LineToExecute>`
		Defines a string that represents the URL or script which is executed along with its parameters.
		The syntax includes:
		• $—Used when you specify an IMO property definition.
		• ~—Used when you specify a Soft Property definition.
		• $number$—The parameter whose index is a number.
		• $USERNAME$—The logged-in username.
		• $GW$ —The Prime Network gateway IP address.
		Note The & character should be written as `&amp;` when used inside a BQL command.
		You can define the launch point of the external application from a soft property of the supported IMO by using the OID of the supported IMO.
		Note The soft property need to be already defined in the IMO of the supported network element.

`MenuCaption / String`	Example: `<MenuCaption type="String">ping</MenuCaption>`	The wording of the right-click menu option to be used for the external launch. **Note** Unless you want to create a mnemonic, do not use an underscore (_) character at the beginning of the string. For example, if `_Scripts` is specified as the menu caption, the result will be having the menu caption as `Scripts`, where `S` is the mnemonic.
`MenuPath / String`	Example: `<MenuPath type="String">external launch demo</MenuPath>`	The menu path to be followed to get to the external launch point. You can have a sub-menu separated by a forward slash (/).
`Parameters / IMObjects_Array`	`<Parameters type="IMObjects_Array">`	All external launch parameters must be passed in an array. See management.IExternalLaunchParameter, page 8-8.

Table 8-1 BQL Command Format and Syntax (continued)

Property / Type	Value	Description
`Role / String`	Example: `<Role type=“string”>CONFIGURATOR</Role>`	Defines the minimum user role required to use the launch point—administrator, configurator, and so on. If this is not defined, all users will have access to the launch point.
`Selection Limit / Int`	Example: `<SelectionLimit type="int">2</SelectionLimit>`	Defines the number of items that must be selected for the launch point to be displayed. For example, if the selection limit is 2, the right-click menu option will not appear if only one item is selected, or if three items are selected. It will only appear if two items are selected.
management.IExternalLaunchParameter		
`ID / OID`	{[ExternalLaunchParameter(Index=1)]}	Index value defines the order in which the parameters will be passed to the script.

`IMOType / String`	Example 1:
	`<IMOType type="String">com.sheer.imo.IPortConnector</IMOType>`
	Example 2:
	`<IMOType type="String">com.sheer.imo.newalarm.ITicket</IMOType>`
		Defines the IMO context type. You can also leave this field empty.
		You can pass the parameter not only from the IMO and its parent on which you set the menu but also from the IMO tree up to IManagedElement. It can be the parent or the higher level IMO depending on the IMO that you have selected for your menu.

`PropertyName / String`	Example 1:
	`<PropertyName type="String">PortAlias</PropertyName>`
	Example 2:
	`<PropertyName type="String">LatestState</PropertyName>`
	Example 3 (retrieving properties of a contained IMO):
	`<PropertyName type="String">ObjectId.Key</PropertyName>`
	Example 4 (retrieving properties not available at time of execution):
	`<PropertyName type="String">Port#PortDescription</PropertyName>`
		Defines the property or soft property name within the IMO context type. This value is used as an argument when invoking the application.
		You can also have any constant value.
		If the property value is another IMO, you can retrieve the properties of this IMO using a dot notation (see Example 3).
		If the property value is an OID, you can retrieve the properties of the referenced IMO using the # notation (see Example 4).

Table 8-1 BQL Command Format and Syntax (continued)

Property / Type	Value	Description
`RunSingleCommand / Boolean`	Example: `<RunSingleCommand type="boolean">true</RunSingleCommand>` The script will be called once, with the required properties of all selected IMOs passed at the same time.	If the value is set to true, a single script will be run for all the selected objects. If set to false, a script will be run for each of the selected objects.
`ReplaceNullWith / String`	Example: `<ReplaceNullWith type="String">N/A</ReplaceNullWith>`	If you want a null value not to be ignored or replaced with an empty string, you can use the ReplaceNullWith string. This is optional. In the example, the null value will be replaced by N/A. See Defining Soft Property Parameters for a BQL Script for more information on the usage of this property.

Defining Soft Property Parameters for a BQL Script

You can use a soft property as a parameter in an external launch BQL command. You can specify the soft property that you want to define the external launch to be performed. To specify this, enter the following in the line to execute property of the BQL Command:

```
<LineToExecute
type="String">http://url.example.com/$GW$/$com.sheer.imo.IEthernetService.NetworkElemnt#~s
oft property~$/$USERNAME$</LineToExecute>
```

To define a soft property parameter you need to write the soft property name (not the label) in the PropertyName entry, and add the following entry to the parameter definition:

```
 <SoftProperty type="boolean">true</SoftProperty>
```

If you do not add this entry, by default the value is assumed to be false.

Parameters that have the value true for the soft property entry are not validated in the definition of the command.

The launch point will be visible even if the IMO does not have the soft property defined on it. When a soft property is not defined, an appropriate message is displayed after the command is run.

Note If you create an ExternalBatchtoExecute-based external launch command that includes a soft property and the specified soft property does not exist on the IMO, the output will have an empty string (" "). This is to ensure that any logic built in to the script which depends on the parameter order, functions as desired. You may use the ReplaceNullWith external launch command parameter if you want to get a different string for the null value.

Running BQL Scripts

See the *Cisco Prime Network Integration Developer Guide* to understand how to run a BQL script.

Enabling Automatic Download of BQL Scripts to Clients

You have the option to have the scripts downloaded automatically to all clients associated with the gateway, upon next login.

To enable automatic download of scripts to clients:

Step 1 Create a directory for your scripts under the Main/webstart/scripts directory on the Prime Network gateway.

Step 2 Run [~/Main/scripts]% updateXLaunchScripts.pl on the gateway to update the auto-deployable scripts jar.

The next time a client connects to the gateway, the scripts will be downloaded and extracted by the client application.

Deleting an External Launch Point

You can delete an external application launch point using the BQL **Delete** command.

Note See the *Cisco Prime Network Integration Developer Guide* for an understanding of BQL, and as a prerequisite to understanding this section.

To delete the launch point:

Step 1 Create a BQL script with the **Delete** command as shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
<command name="Delete">
    <param name="oid">
        <value>
            <management.IExternalLaunch>
               <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnector
               )(Name=example4)]}</ID>
            </management.IExternalLaunch>
        </value>
    </param>
</command>
```

Step 2 Run the BQL script.

Step 3 Launch Prime Network Vision to verify that the launch point has been removed.

Sample BQL Scripts for Launching External Applications

This chapter provides example scripts for launching external applications using BQL.

Note See Table 8-1 on page 8-4 for details on the syntax used in the commands.

For a BQL Example That:	See
Runs a single script on selected tickets using their latest state and last modification time as parameters	Example 1, page 8-11
Runs a single script on selected tickets that include Layer 2 in their description, using their latest state and last modification time as parameters	Example 2, page 8-12
Includes a soft property parameter in a BQL command to create an external launch point on a network element	Example 3, page 8-13
Uses a batch file to invoke an application	Example 4, page 8-13
Executes a script named C:/script.batch using PortAlias and the logged-in username as parameters	Example 5, page 8-14
Launches a URL that uses the parameters values gateway IP, NE IP and the logged-in username; and uses / as a delimiter	Example 6, page 8-14
Launches a URL that uses the parameters a, b, and c whose values are, respectively, the soft property showClock, a constant string "constant", and the logged-in username	Example 7, page 8-15
Launches a URL that uses the parameters a, b, c and d whose values are, respectively, the IMO property IP, the property PortAlias, a constant string "any_const_parameter", and the logged-in username; and uses the existing Parameters element	Example 8, page 8-15
Uses an environment variable to invoke an application	Example 9, page 8-16
Uses multiple IMO contexts to launch an application	Example 10, page 8-16

For details about the format an d the syntax of BQL scripts, see Creating BQL Scripts, page 8-4.

Note `<param name = “replace”>` is always set to **`true`**. This value replaces the old value with the new value based on the script that is invoked.

Example 1

This example runs a single script on the selected tickets with their latest state and last modification time as parameters.

```
<command name="Set">
  <param name="imo">
    <value>
      <management.IExternalLaunch>

             <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.newalarm.ITicket
             )(Name=print state)]}</ID>
           <ExternalBatchToExecute type="String">C:\script.bat</ExternalBatchToExecute>
           <MenuCaption type="String">Print state</MenuCaption>
           <MenuPath type="String">My apps</MenuPath>
           <Parameters type="IMObjects_Array">
           <management.IExternalLaunchParameter>
                   <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</ID>
                   <IMOType type="String">com.sheer.imo.newalarm.ITicket</IMOType>
                   <PropertyName type="String">LatestState</PropertyName>
           </management.IExternalLaunchParameter>
           <management.IExternalLaunchParameter>
                   <ID type="Oid">{[ExternalLaunchParameter(Index=2)]}</ID>
                   <IMOType type="String">com.sheer.imo.newalarm.ITicket</IMOType>
                   <PropertyName type="String">LastModificationTime</PropertyName>
           </management.IExternalLaunchParameter>
           </Parameters>
           <RunSingleCommand type="boolean">true</RunSingleCommand>
       </management.IExternalLaunch>
    </value>
  </param>
  <param name="replace">
    <value>true</value>
  </param>
</command>
```

Example 2

This example runs a single script on the selected tickets with their latest state and last modification time as parameters, for tickets that include Layer 2 in their description.

```
<command name="Set">
  <param name="imo">
    <value>
        <management.IExternalLaunch>

            <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.newalarm.ITicket)
            (Name=print state)]}</ID>
           <ExternalBatchToExecute type="String">C:\script.bat</ExternalBatchToExecute>
                    <LatestState type="String">Layer 2</LatestState>
           <MenuCaption type="String">Print state</MenuCaption>
           <MenuPath type="String">My apps</MenuPath>
           <Parameters type="IMObjects_Array">
           <management.IExternalLaunchParameter>
                   <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</ID>
                   <IMOType type="String">com.sheer.imo.newalarm.ITicket</IMOType>
                   <PropertyName type="String">LatestState</PropertyName>
           </management.IExternalLaunchParameter>
           <management.IExternalLaunchParameter>
                   <ID type="Oid">{[ExternalLaunchParameter(Index=2)]}</ID>
                   <IMOType type="String">com.sheer.imo.newalarm.ITicket</IMOType>
                   <PropertyName type="String">LastModificationTime</PropertyName>
           </management.IExternalLaunchParameter>
           </Parameters>
           <RunSingleCommand type="boolean">true</RunSingleCommand>
       </management.IExternalLaunch>
    </value>
  </param>
  <param name="replace">
    <value>true</value>
  </param>
</command>
```

Example 3

This example includes a soft property parameter in a BQL command for an external launch point on a network element.

```
<command name="Set">
    <param name="imo">
        <value>
            <management.IExternalLaunch>

                <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IManagedEleme
                 nt)(Name=InfoVista-Launch-test-2)]}</ID>
                <ElementType type="String">Cisco 7206VXR</ElementType>
                <ExternalBatchToExecute
                 type="String">C:/IV/launch-report.bat</ExternalBatchToExecute>
                <MenuCaption type="String">test4</MenuCaption>
                <MenuPath type="String">Performance/Test4</MenuPath>
                <Parameters type="IMObjects_Array">
                    <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</ID>
                        <IMOType type="String">com.sheer.imo.IManagedElement</IMOType>
                        <PropertyName type="String">showClock</PropertyName>
                        <SoftProperty type="boolean">true</SoftProperty>
                    </management.IExternalLaunchParameter>
                    <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=2)]}</ID>
                        <IMOType type="String">com.sheer.imo.IManagedElement</IMOType>
                        <PropertyName type="String">IP</PropertyName>
                        <SoftProperty type="boolean">false</SoftProperty>
                    </management.IExternalLaunchParameter>
                    <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=3)]}</ID>
                        <IMOType type="String">com.sheer.imo.IManagedElement</IMOType>
                 <PropertyName type="String">ElementTypeKey</PropertyName>
                    </management.IExternalLaunchParameter>
                </Parameters>
                <Vendor type="String">Cisco</Vendor>
            </management.IExternalLaunch>
        </value>
    </param>
    <param name="replace">
   <value>true</value>
    </param>
</command>
```

Example 4

This example uses a batch file to invoke an application.

```
<command name="Set">
    <param name="imo">
        <value>
            <management.IExternalLaunch>

                <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IManagedEleme
                 nt)(Name=example1)]}</ID>
                <ElementType type="String">Cisco</ElementType>
                <Vendor type="String">Cisco</Vendor>
                <ExternalBatchToExecute
                 type="String">C:\runthis.bat</ExternalBatchToExecute>
                <MenuCaption type="String">ping</MenuCaption>
                <MenuPath type="String">external launch/file example</MenuPath>
                <Parameters type="IMObjects_Array">
                    <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</ID>
                        <IMOType type="String">com.sheer.imo.IManagedElement</IMOType>
                        <PropertyName type="String">IP</PropertyName>
                    </management.IExternalLaunchParameter>
                </Parameters>
            </management.IExternalLaunch>
        </value>
    </param>
    <param name="replace">
        <value>true</value>
    </param>
</command>
```

Example 5

This example executes a script `C:/script.batch` with parameters PortAlias and the logged-in username.

```
<?xml version="1.0" encoding="UTF-8"?
<command name="Set
   <param name="imo">
      <value>
         <management.IExternalLaunch>
            <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnector)
               (Name=example)]}</ID
             <MenuCaption type="String">My Example</MenuCaption>
             <MenuPath type="String">My Menu/My Sub Menu</MenuPath>
             <LineToExecute type="String">c:/script.batch
              $com.sheer.imo.IPortConnector.PortAlias $USERNAME$</LineToExecute>
         </management.IExternalLaunch>
     </value>
   </param>
  <param name="replace">
    <value>true</value>
  </param>
</command>
```

Example 6

This example launches a URL with parameter values Gateway IP, the NE IP and the logged-in username, using / (forward slash) as delimiter.

```
<?xml version="1.0" encoding="UTF-8"?
<command name="Set">
    <param name="imo">
        <value>
            <management.IExternalLaunch>
              <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnector)
               (Name=example)]}</ID>
              <ElementType type="String">IOU</ElementType>
              <Vendor type="String">Cisco</Vendor>
              <MenuCaption type="String">My Example</MenuCaption>
              <MenuPath type="String">My Menu/My Sub Menu</MenuPath>
              <LineToExecute type="String">http://url.example.com/$GW$/$com.sheer.imo.IMan
               agedElement.IP$/$USERNAME$</LineToExecute>
            </management.IExternalLaunch>
        </value>
    </param>
    <param name="replace">
        <value>true</value>
    </param>
</command>
```

Example 7

This example launches a URL with parameters a, b, and c, whose values are the soft property `showClock`, a constant string `constant`, and the logged-in username.

```
<?xml version="1.0" encoding="UTF-8"?>
<command name="Set">
    <param name="imo">
        <value>
            <management.IExternalLaunch>
              <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnector)
               (Name=example)]}</ID>
              <ElementType type="String">IOU</ElementType>
              <MenuCaption type="String">My Example</MenuCaption>
              <MenuPath type="String">My Menu/My Sub Menu</MenuPath>
              <LineToExecute type="String">http://url.example.com/my.action?a=~com.sheer.i
               mo.IManagedElement.showClock~&amp;b=constant&amp;c=$USERNAME$</LineToExecut
               e>
            </management.IExternalLaunch>
        </value>
    </param>
    <param name="replace">
       <value>true</value>
    </param>
</command>
```

Example 8

This example launches a URL with parameters a, b, c, and d, whose values are the IMO property IP, property PortAlias, a constant string `any_const_parameter`, and the logged-in username, using the existing Parameters element.

```
<?xml version="1.0" encoding="UTF-8"?>
<command name="Set">
   <param name="imo">
        <value>
            <management.IExternalLaunch>
              <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnector)
               (Name=example)]}</ID>
              <ElementType type="String">IOU</ElementType
              <MenuCaption type="String">My Example</MenuCaption>
              <MenuPath type="String">My Menu/My Sub Menu</MenuPath
              <LineToExecute type="String">http://url.example.com/my.action?a=$1$&amp;b=$2
               $&amp;c=$3$&amp;d=$USERNAME$</LineToExecute>
              <Parameters type="IMObjects_Array">
                 <management.IExternalLaunchParameter>
                   <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</ID>
                   <IMOType type="String">com.sheer.imo.IManagedElement</IMOType>
                   <PropertyName type="String">IP</PropertyName>
                 </management.IExternalLaunchParameter>
                 <management.IExternalLaunchParameter>
                   <ID type="Oid">{[ExternalLaunchParameter(Index=2)]}</ID>
                   <IMOType type="String">com.sheer.imo.IPortConnector</IMOType>
                   <PropertyName type="String">PortAlias</PropertyName>
                 </management.IExternalLaunchParameter>
                 <management.IExternalLaunchParameter>
                   <ID type="Oid">{[ExternalLaunchParameter(Index=3)]}</ID>
                   <IMOType type="String"></IMOType>
                   <PropertyName type="String">any_const_parameter</PropertyName>
                 </management.IExternalLaunchParameter>
              </Parameters>
             </management.IExternalLaunch>
       </value>
    </param>
    <param name="replace">
       <value>true</value>
    </param>
</command>
```

Example 9

This examples uses an environment variable to invoke an application.

```
<command name="Set">
    <param name="imo">
        <value>
            <management.IExternalLaunch>

                <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IManagedEleme
                 nt)(Name=example3)]}</ID>
                <ElementType type="String">Cisco</ElementType>
                <Vendor type="String">Cisco</Vendor>
                <ExternalBatchToExecute
                 type="String">%EMS_A%runthis.bat</ExternalBatchToExecute>
                <MenuCaption type="String">environment variable</MenuCaption>
                <MenuPath type="String">External Launch</MenuPath>
                <Parameters type="IMObjects_Array">
                    <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</ID>
                        <IMOType type="String"></IMOType>
                        <PropertyName type="String">www.cisco.com</PropertyName>
                    </management.IExternalLaunchParameter>
                   <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=2)]}</ID>
                        <IMOType type="String">com.sheer.imo.IManagedElement</IMOType>
                        <PropertyName type="String">IP</PropertyName>
                    </management.IExternalLaunchParameter>
                </Parameters>
            </management.IExternalLaunch>
        </value>
    </param>
    <param name="replace">
        <value>true</value>
    </param>
</command>
```

Example 10

This example uses multiple IMO contexts to launch an application.

```
<command name="Set">
    <param name="imo">
        <value>
            <management.IExternalLaunch>

                <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
                 r)(Name=example4)]}</ID>
                <ElementType type="String">IOU</ElementType>
                <Vendor type="String">Cisco</Vendor>
                <ExternalBatchToExecute
                 type="String">c:runthis.bat</ExternalBatchToExecute>
                <MenuCaption type="String">ping</MenuCaption>
                <MenuPath type="String">External Launch</MenuPath>
                <Parameters type="IMObjects_Array">
                    <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</ID>
                        <IMOType type="String">com.sheer.imo.IManagedElement</IMOType>
                        <PropertyName type="String">IP</PropertyName>
                    </management.IExternalLaunchParameter>
                    <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=2)]}</ID>
                        <IMOType type="String">com.sheer.imo.IPortConnector</IMOType>
                        <PropertyName type="String">PortAlias</PropertyName>
                    </management.IExternalLaunchParameter>
                    <management.IExternalLaunchParameter>
                        <ID type="Oid">{[ExternalLaunchParameter(Index=3)]}</ID>
                        <IMOType type="String"></IMOType>
                        <PropertyName type="String">any_const_parameter</PropertyName>
                    </management.IExternalLaunchParameter>
                </Parameters>
            </management.IExternalLaunch>
        </value>
    </param>
    <param name="replace">
        <value>true</value>
    </param>
</command>
```

PART 6

Appendixes

These topics provide additional reference and legacy material.

- Soft Properties: Supported Parsing Operators, Rules, and Regular Expressions, page A-1
- Command Manager and Command Builder: Macro Language and Beanshell Reference, page C-1
- VCB: CLI Commands and Template Reference, page B-1

APPENDIX A

Soft Properties: Supported Parsing Operators, Rules, and Regular Expressions

These topics provide listings of the operators, rules, and expressions that are supported for creating soft properties.

- Supported Parsing Rules and Operators, page A-1
- Supported Regular Expressions, page A-3

For information on supported triggers for TCAs, see Creating a New Threshold-Crossing Alarm, page 7-1.

Supported Parsing Rules and Operators

Table A-1 lists the predefined text manipulation operators available for parsing raw device input and turning each into a soft property that is available in the Add/Edit Parsing Rule dialog box.

Table A-1 Supported Parsing Operators and Rules for Soft Properties

Operator	Description	Parameter	Parameter Description	Parameter Validation Rules
Header and Footer	Removes a specified number of lines from the header and footer of the input text.	Header lines	Number of header lines to be removed.	Integers only. Mandatory.
		Footer lines	The number of footer lines to be removed.	Integers only. Mandatory.
Remove Lines	Removes a range of lines from the specified starting row to the specified end row the input text.	From line	Index of first row to begin removal, inclusive.	Integer only. Mandatory.
		To line	Index of last row to be removed, inclusive.	Integer only; must be equal to or greater than *From line*. Mandatory.
Select Lines	Extracts a range of lines from the specified starting row to the specified end row f the input text.	From line	Index of first row to begin selection, inclusive.	Integer only. Mandatory.
		To line	Index of last row to be selected, inclusive.	Integer only; must be equal to or greater than *From line*. Mandatory.

Table A-1 Supported Parsing Operators and Rules for Soft Properties (continued)

Operator	Description	Parameter	Parameter Description	Parameter Validation Rules
Replace	Finds one substring or all occurrences of a substring that matches a specified regular expression, and replaces it with a specified value.	Expression	Search for value or regular expression.	Text. Mandatory.
		With	Replace string with value or regular expression.	Text. Mandatory.
		From Index	Starting index.	Integer; must be 1 or higher. Mandatory.
		Replace All	Check box. Select this option to replace all occurrences of the matching substrings, otherwise only the first instance is replaced.	Default is unchecked.
Match	Finds and extracts a substring that matches a specified regular expression. If no match can be found, the output buffer receives an empty string or receives the original input, based on choice in *If no match* drop down list.	Expression	Search for value or regular expression.	Text. Mandatory.
		If no match	Returns the original input or and empty string, based on your choice.	
Set	Prints the results of the input and output buffers.	Expression	Regular expression template to use for formatting. $_$ specifies the main output buffer.	Text. Mandatory.
Substring	Extracts a substring of a specified length from a specified starting point.	From Index	Begin index to select.	Integer; must be 1 or higher. Mandatory.
		Length	How many characters to select.	Integer. Mandatory.
Parse Integer	Uses the substring rule. When a result is received with the substring, it is converted into an integer value. **Note** If the substring operator contains any characters, the parsing integer operator fails.	From Index	Starting index to select.	Integer; must be 1 or higher. Mandatory.
		To Index	Ending index to select.	Integer. Mandatory.

Supported Regular Expressions

A regular expression consists of a character string in which some characters are given special meaning with regard to pattern matching. These topics are based on the documentation of the package GNU RegExp. These tables list the supported and unsupported expressions.

Note Some flavors of regular expression utilities support additional escape sequences. The following is not meant to be an exhaustive list. In the future, gnu.regexp might support some or all of the following but they are currently unsupported:

(?mods) — Inline compilation/execution modifiers (Perl5)

\G — End of previous match (Perl5)

[.*symbol*.] — Collating symbol in class expression (POSIX).

- **[=*class*=] —** Equivalence class in class expression (POSIX)
- **s/foo/bar —** Style expressions as in sed and awk

Table A-2 Supported Positional Operators

Character	Description
^	Matches at the beginning of a line.
$	Matches at the end of a line.
\A	Matches the start of the entire string.
\Z	Matches the end of the entire string.
\b	Matches at a word break (Perl5 syntax only).
\B	Matches at a nonword break (opposite of \b) (Perl5 syntax only).
\<	Matches at the start of a word (egrep syntax only).
\>	Matches at the end of a word (egrep syntax only).

Table A-3 Supported One-Character Operators

Character	Description
.	Matches any single character.
\d	Matches any decimal digit.
\D	Matches any nondigit.
\n	Matches a newline character.
\r	Matches a return character.
\s	Matches any whitespace character.
\S	Matches any nonwhitespace character.
\t	Matches a horizontal tab character.
\w	Matches any word (alphanumeric) character.

Table A-3 Supported One-Character Operators (continued)

Character	Description
\W	Matches any nonword (alphanumeric) character.
\x	Matches the character *x*, if *x* is not one of the above listed escape sequences.

Table A-4 Supported Character Class Operators

Character	Description
[*abc*]	Matches any character in the set *a*, *b*, or *c*.
[^*abc*]	Matches any character not in the set *a*, *b*, or *c*.
[*a*-*z*]	Matches any character in the range *a* to *z*, inclusive.
Leading or trailing dash	Interpreted literally.

Table A-5 Supported Character Class Operators—if RE_CHAR_CLASSES Is Enabled

Character Sequences	Description
[:alnum:]	Any alphanumeric character.
[:alpha:]	Any alphabetic character.
[:blank:]	A space or horizontal tab.
[:cntrl:]	A control character.
[:digit:]	A decimal digit.
[:graph:]	A nonspace, noncontrol character.
[:lower:]	A lowercase letter.
[:print:]	Same as graph, but also space and tab.
[:punct:]	A punctuation character.
[:space:]	Any whitespace character, including newline and return.
[:upper:]	An uppercase letter.
[:xdigit:]	A valid hexadecimal digit.

Table A-6 Supported Subexpressions and Back References

Expression	Description
(*abc*)	Matches whatever the expression *abc* would match, and saves it as a subexpression. Also used for grouping.
(?:...)	Pure grouping operator; does not save contents.
(?#...)	Embedded comment; ignored by engine.
\n	Where 0 < *n* < 10, matches the same thing the *n*th subexpression matched.

Table A-7 Supported Branching (Alternation) Operator

Expression	Description
a|*b*	Matches whatever the expression *a* or *b* would match.

Table A-8 Supported Repeating Operators (Operate on Previous Atomic Expression)

Symbols	Description
Note	If any of these operators are immediately followed by a question mark (?), the repeating operator stops at the smallest number of repetitions that can complete the rest of the match (Stingy (Minimal) Matching).
?	Matches the preceding expression or the null string.
*	Matches the null string or any number of repetitions of the preceding expression.
+	Matches one or more repetitions of the preceding expression.
{*m*}	Matches exactly *m* repetitions of the one-character expression.
{*m*,*n*}	Matches between *m* and *n* repetitions of the preceding expression, inclusive.
{*m*,}	Matches *m* or more repetitions of the preceding expression.

Table A-9 Supported Lookahead

Expression	Description
Note	Lookahead refers to the ability to match part of an expression without consuming any of the input text. The variations in this table are supported.
(?=*foo*)	Matches at any position where *foo* would match, but does not consume any characters of the input.
(?!*foo*)	Matches at any position where *foo* would not match, but does not consume any characters of the input.

APPENDIX B

VCB: CLI Commands and Template Reference

These topics provide reference information for VCB CLI commands and templates:

- VCB CLI Reference: vcb and vcb sitechanges Commands, page B-1
- VCB CLI Reference: U-VNEs Commands, page B-3
- VCB CLI Reference: Standard and Pluggable Modules Commands, page B-17
- VCB CLI Reference: Events Commands, page B-31
- VCB Template Reference, page B-56

VCB CLI Reference: vcb and vcb sitechanges Commands

These topics provide reference information for the **vcb** and **vcb sitechanges** commands which are used to deploy, maintain, and remove VCB customizations.

- vcb, page B-1
- vcb sitechanges, page B-3

vcb

The **vcb** command is the wrapper for all VCB-related commands.

Syntax

```
{uvne|module|pluggablemodule|event|eventpattern|eventparsingrules|devicetype}
{add|view|modify|delete} args… [-help ] [-debuglevel {error|warn|info|debug}] [-logfile
logfile] -user username -password password

vcb {eventarg} {view} [-help ] [-debuglevel {error|warn|info|debug}] [-logfile logfile]]
-user username -password password

vcb {sitechanges} {view|export|delete} [-help ] [-debuglevel {error|warn|info|debug}]
[-logfile logfile]] -user username -password password
```

For information—descriptions, options, and usage—about specific commands, see specific command references:

- VCB CLI Reference: U-VNEs Commands, page B-3
- VCB CLI Command Reference: Device Types Commands, page B-12

- VCB CLI Command Reference: Standard Modules, page B-18
- VCB CLI Command Reference: Pluggable Modules, page B-25

Global Command Options

Table B-1 describes the global options and arguments that are common to all **vcb** commands and subcommands.

Table B-1 Global Options and Arguments—vcb

Option ***Argument***	**Description**
help	Displays online help about the command.Use this option for each subcommand.
debuglevel	Determines which messages are logged based on the message severity. Valid values are: error, warn (default), info, and debug. For example, if debuglevel is set to warn, all warning and error messages are saved to the log file.
logfile *logfile*	Logs the CLI output to the file specified in the *logfile* argument.
userdefined	Displays the registrations that have been added to the site.xml file using the VCB or any other tool.
user *username*	BQL username.
password *password*	BQL password.

Description

Use the **vcb** command to create and update VNE registry configuration files. Use the **vcb** command to add, delete, view, and modify information in these files, based on the subcommands that are used with it.

The **vcb** command performs the following high-level operations:

- Executes Prime Network administrator-level operations. It also provides a service to handle authorization errors. VCB users must have the Prime Network admin user role and associated privileges in order to perform registry configuration commands.
- Verifies that the expected version of Prime Network is running.
- Provides help information for each command.
- Logs all internal commands.
- Centralizes error handling for exceptions.

General Error Codes

Table B-2 General Error Codes—vcb

Code	**Description**
0	Signifies OK.
10	Operation not permitted. An attempt was made to perform an operation limited to processes with appropriate privileges or to the owner of a file or other resources.
11	User does not have Prime Network admin privileges.

Table B-2 General Error Codes—vcb (continued)

Code	Description
20	No such file or directory. A component of a specified pathname did not exist, or the pathname was an empty string. This might occur when the argument is a DSP or to a registry hive.
30	Failed to connect to Prime Network server.
40	I/O error. Some physical input or output error occurred.
50	Invalid argument. Some invalid argument was supplied.
60	Incompatible Prime Network version of VCB; expected $vcb_expected_version$ and found Prime Network $ana_running_version$.
70	Bad procedure for program. A Remote Procedure Call (RPC) call was attempted for a procedure which does not exist in the VCB program.
80	Function not implemented. Attempted a system call that is not available on this system.

vcb sitechanges

The **vcb sitechanges** command affects all extensions in the local site.xml file—displaying, exporting, or deleting them—whether the extensions were created by the VCB or by other Prime Network utilities.

Syntax

```
vcb sitechanges {view|export|delete} [-help ] [-debuglevel {error|warn|info|debug}]
[-logfile logfile]] -user username -password password
```

Description

Use the **vcb sitechanges** command to view all customizations that were made to the site.xml registry file whether by the VCB or by other Prime Network utilities. Use the **vcb sitechanges** command also to write all customizations to script files that enable you to import changes to another system or to delete all changes, returning the system to factory default.

Because the VCB does not differentiate between changes made by the VCB and changes made by other Prime Network utilities, the **vcb sitechanges export** and **vcb sitechanges delete** commands create script files to enable you to inspect commands before you execute them. Edit the script files before running them to ensure that only the customizations that you are interested in are acted upon.

VCB CLI Reference: U-VNEs Commands

These topics provide reference information for the **vcb uvne** commands you can use to create, view, modify, and delete U-VNEs:

- vcb uvne add, page B-4
- vcb uvne view, page B-6
- vcb uvne modify, page B-9
- vcb uvne delete, page B-11

vcb uvne add

The **vcb uvne add** command creates a U-VNE by associating a U-VNE template with the device type associated with the given sysOID, or by cloning from an existing device or device family.

Syntax

```
vcb uvne add -sysoid sysoid -template template name -group template filename [-devicetype
device type name] -user username -password password
```

```
vcb uvne add -sysoid sysoid -clonesysoid clone_sysoid -user username -password password
```

```
vcb uvne add -sysoid sysoid -clonedevicefamily devicefamily [-devicetype device type]
-user username -password password
```

```
vcb uvne add -sysoid sysoid -softwareversion software-version -clonesoftwareversion
clone-softwareversion -clonedevicefamily devicefamily -scheme <product|ipcore >
[-devicetype device type name] -user username -password password
```

Description

The **vcb uvne add** command creates a U-VNE-driver for the device type associated with the given sysOID using the specified U-VNE template. Creating a U-VNE-driver enables the Auto Detect feature in Prime Network to associate a device with this sysOID with the VNE-driver implementation defined by the template.

This command creates a separate registry configuration for the U-VNE. Any configuration setting given in the command parameters affect this copy.

Note This command does not create the device category for the device. For more information, see vcb devicetype add, page B-12.

Usage Examples

Example 1

```
vcb uvne add -sysoid .1.2.3.4 -template GenericUVNE -group uvne -user root -password admin
```

Enables discovery of the device with sysOID 1.2.3.4 using the GenericUVNE template, which is located in group uvne-product.

The command does not create any device attributes for the newly added device. To assign the device a user-friendly name and the correct device category, use the **-devicetype** option (see the "vcb devicetype add" section on page B-12).

Example 2

```
vcb uvne add -sysoid .1.2.3.4 -clonesysoid .1.3.6.1.4.1.9.1.108 -user root -password
admin
```

Enables discovery of a new device (with sysOID .1.2.3.4) that points to registration—scheme and instrumentation commands—of the already supported sysOID .1.3.6.1.4.1.9.1.108.

(To create a list of already supported VNEs, see vcb uvne view, page B-6.)

Example 3

```
vcb uvne add -sysoid .1.2.3.4 -softwareversion 12.6(2) -clonesoftwareversion 12.0(23)S2
-clonedevicefamily 100xx -scheme product -user root -password admin
```

Adds support for a new device (with sysOID .1.2.3.4) based on the device family 100xx. Also adds support for the new software version for this device family. (For a list of already supported software versions, see vcb uvne view, page B-6.)

Example 4

```
vcb uvne add -sysoid .1.2.3.4 -clonedevicefamily 100xx -user root -password admin
```

Enables discovery of a new device (with sysOID .1.2.3.4), that points to registrations—scheme and instrumentation commands—for the device family 100xx.

Options

Table B-3 Options and Arguments—vcb uvne add

Option *Argument*	Description
sysoid *sysoid*	The sysObject ID of the device for which to create a U-VNE-driver using the implementation defined in the U-VNE template. **Note** Each sysOID value in the system must be unique.
template *template name*	The name of the U-VNE template from which to create the U-VNE-driver. See U-VNE Templates, page B-56. **Note** This option is mutually exclusive with the **-clonesysoid** and **-clonedevicefamily** options.
group *template filename*	The name of the file in which the U-VNE template is located. U-VNE templates are located in the uvne-product file. **Note** Use this option with the **-template** option only.
devicetype *device type name*	(Optional) The U-VNE device type name. If not specified, the device type is: • Defined as Unknown when the option is not specified. • Inherited from the device or device family when you use the **-clonesysoid** option. The device type name appears in the UI and is associated with a category; its category determines which icon is displayed, and other presentation aspects are derived from this reference. To use a new device type name, first add it using the **vcb devicetype add** command. For more information, see vcb devicetype add, page B-12.
clonesysoid *clone-sysoid*	The sysObject ID of an already supported VNE. To view sysOIDs for supported VNEs, use the **vcb uvne view -sysoid** all command.
softwareversion *software-version*	The software version that you want to add for a U-VNE. To obtain the software version string, use the **show version** command.
clonesoftwareversion *clone-softwareversion*	An already supported software version that you want to clone. The software version must already be supported for a particular device family. For a list of supported software versions, use the command **vcb uvne view -scheme <***ipcore|product***> -devicefamily <***device family***> -user** *user* **-password** *password***.** For more information, see vcb uvne view, page B-6.

Table B-3 Options and Arguments—vcb uvne add (continued)

Option ***Argument***	**Description**
clonedevicefamily *devicefamily*	A supported device family. For a list of supported device families, use the command **vcb uvne view –scheme** <*ipcore*|*product*> **–devicefamily** *devicefamily* **–user** *user* **-password** *password*. For more information, see vcb uvne view, page B-6. **Note** This option is mutually exclusive with the **-template** option.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-4 Error Codes—vcb uvne add

Code	**Description**
101	The sysOID already exists and is already modeled as a VNE.
102	U-VNE template file not found.
103	No such template name in the templates file.
104	No such device type configuration exists.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb uvne view

The **vcb uvne view** command returns an existing U-VNE configuration.

Syntax

```
vcb uvne view -sysoid {sysoid | all} -user username -password password
[-userdefined] [-detail]
```

```
vcb uvne view -template {template name| all} -group <template group name> -user username
-password password
```

Description

Use the **vcb uvne view** command to:

- Display information based on the specified sysOID.
- Display available templates found in the specified group.

Usage Examples

Example 1

```
vcb uvne view -sysoid .1.2.3.4 -user root -password admin -userdefined
```

Returns the configuration of the VNE with the sysOID 1.2.3.4, including the U-VNE template (or the device family for a developed VNE). If device-type associations are defined (using the **vcb devicetype add** command), these associations are also displayed.

Example 2

```
vcb uvne view -sysoid all -user root -password admin
```

Returns all the sysoid supported in Prime Network.

Example 3

```
vcb uvne view -sysoid all -user root -password admin -detail
```

Returns all known configured sysOIDs (regular VNEs) and also this command would show following additional informations

- Module Spec Name associated with the sysoid.
- Syslog and Trap Parsing rule name associated with the sysoid

Here is an example of the output:

```
SysOid......:.9.7.6.5.4.3.1
DeviceFamily:76xx
CloneSysOid.:.1.3.6.1.4.1.9.1.863
Scheme........:product
Module Spec.........:ciscophysicalspec2
Trap Parsing Rule...:cisco-trap-product-parsing-rules
Syslog Parsing Rule.:cisco-syslog-product-parsing-rules
Scheme........:ipcore
Module Spec.........:ciscophysicalspec2
Trap Parsing Rule...:cisco-trap-ipcore-parsing-rules
Syslog Parsing Rule.:cisco-syslog-ipcore-parsing-rules

SysOid.....:.3.4.5.6.7
Template...:GenericUVNE
Group......:uvne
Device Type:UNKNOWN
Scheme........:product
Module Spec.........:N/A
Trap Parsing Rule...:genericuvne-trap-parsing-rules
Syslog Parsing Rule.:genericuvne-syslog-parsing-rules
```

Options

Table B-5 Options and Arguments—vcb uvne view

Option *Argument*	Description
sysoid *sysoid*	Returns the configuration of the specified VNE, including the device type, the user-friendly name, and the template (for U-VNEs). **Tip** Enter **all** as the *sysoid* to view the configuration of all sysOIDs (U-VNEs and developed VNEs) configured in the system.
group *template filename*	Returns the configuration of all sysOIDs and templates contained in the specified group. **Tip** Enter **all** as the *template filename* to view the template associations for each group in the system.
userdefined	Lists the U-Vne created through vcb command only.
details	Shows following additional information: • Module Spec Name associated with the sysoid. • Syslog and Trap Parsing rule name associated with the sysoid

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-6 Error Codes—vcb uvne view

Code	Description
102	U-VNE template file not found.
103	No such template name in the templates file.
104	No such device type configuration exists.
111	The SysOID specified does not exist.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb uvne modify

The **vcb uvne modify** command modifies the configuration of an existing U-VNE.

Syntax

```
vcb uvne modify -sysoid sysoid -template template name -group template filename -user
username -password password
```

```
vcb uvne modify -sysoid sysoid -devicetype device type -user username -password password
```

```
vcb uvne modify -sysoid sysoid -clonesysoid sysoid -user username -password password
```

```
vcb uvne modify –sysoid sysoid -softwareversion softwareversion
-clonesoftwareversion clonesoftwareversion -clonedevicefamily devicefamily -scheme
vnescheme -user username -password password
```

```
vcb uvne modify –sysoid sysoid -clonedevicefamily devicefamily -user username -password
password
```

Description

Use the **vcb uvne modify** command to:

- Associate the U-VNE with another U-VNE template.
- Change the device type associated with the U-VNE.
- Change the cloning reference of a U-VNE.

Usage Examples

Example 1

```
vcb uvne modify -sysoid .1.2.3.4 -group uvne-product -template GenericUVNE -user root
-password admin
```

Modifies the template of the U-VNE to the newly specified template defined in the given group.

Example 2

```
vcb uvne modify -sysoid .1.2.3.4 -devicetype CISCO_1760 -user root -password admin
```

Modifies the device type for the U-VNE with sysOID 1.2.3.4.

Example 3

```
vcb uvne modify -sysoid .1.2.3.4 –clonesysoid .1.3.6.1.4.1.9.1.108 -user root -password
admin
```

Modifies a device, with sysOID .1.2.3.4, to point to a new device family based on the clone sysoid .1.3.6.1.4.1.9.1.108.

Example 4

```
vcb uvne modify -sysoid .1.2.3.4 –clonedevicefamily 12xxx -user root -password admin
```

Modifies support for a new device, with sysOID .1.2.3.4, to point to registrations—scheme and instrumentation commands—of the device family 12xxx.

Options

Table B-7 Options and Arguments—vcb uvne modify

Option ***Argument***	**Description**
sysoid *sysoid*	The sysObject ID of the U-VNE-driver configuration that you want to modify.
template *template name*	The name of the U-VNE template to which the U-VNE should be associated. Use this option to modify the U-VNE template from which the U-VNE-driver derives its configuration. **Note** Using this option does not overwrite other configuration changes made with the VCB, such as user-experience attributes that are defined with the vcb devicetype add command.
group *template filename*	The name of the group that includes the U-VNE template, such as uvne-product.
devicetype *device type*	The device type associated with the U-VNE.
clonesysoid *clone-sysoid*	The sysObject ID of an already supported VNE. To view sysOIDs for supported VNEs, use the **vcb uvne view -sysoid** all command.
softwareversion *software-version*	The software version that you want to support for a U-VNE. To obtain the software version string, use the **show version** command.
clonesoftwareversion *clone-softwareversion*	An already supported software version that you want to clone. The software version must already be supported for a particular device family.
clonedevicefamily *deviceFamily*	A supported device family. For a list of supported device families, use the command **vcb uvne view –sysoid all –user** *user* **-password** *password*. For more information, see vcb uvne view, page B-6.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-8 Error Codes—vcb uvne modify

Code	**Description**
102	U-VNE template file not found.
103	No such template name in the templates file.
104	No such device type configuration exists.
112	The sysOID does not exist or already exists and is already modeled as a VNE.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb uvne delete

The **vcb uvne delete** command deletes a U-VNE.

Syntax

```
vcb uvne delete -sysoid sysoid -user username -password password
```

```
vcb uvne delete -sysoid sysoid -devicefamily DeviceFamilyName -scheme schemeName
-softwareversion "softwareVersionNumber" -user username -password password
```

Description

The **vcb uvne delete** command is useful when migrating from a U-VNE to a developed VNE. If, in an upgrade, Prime Network provides a DSP that contains a developed VNE to support the device type, the need for the U-VNE is eliminated. You must delete the U-VNE before Prime Network can use the developed VNE to model and manage the device.

Deleting a template-based U-VNE has no effect on the U-VNE template from which it derives its implementation.

Usage Example

Example 1

```
vcb uvne delete -sysoid .1.2.3.4 -user root -password admin
```

Deletes the U-VNE-driver configured for the device with sysOID 1.2.3.4.

Example 2

```
vcb uvne delete -sysoid .1.2.3.4 -devicefamily 70xx -scheme product
-softwareversion "12.0(23)S3"
```

Deletes the software configuration for U-VNE driver configured for the device with sysOID .1.2.3.4. Use the command in Example 1 to delete the U-VNE.

Note that the vcb uvne delete syntax should match the vcb uvne add syntax to avoid items being left in the site.xml after the delete action. For example, if the vcb uvne add syntax is as follows:

```
vcb uvne add -sysoid .1.3.6.1.4.1.9.1.917 -softwareversion "15.0(2)SG1"
-clonesoftwareversion "gt 12.2(52)SG" -clonedevicefamily cisco-catalyst-4900-series
-scheme product -devicetype CISCO_CATALYST_4900M -override -user root -password admin
```

then, the vcb uvne delete syntax should be as follows:

```
vcb uvne delete -sysoid .1.3.6.1.4.1.9.1.917 -devicefamily cisco-catalyst-4900-series
-scheme product -softwareversion "15.0(2)SG1" -user root -password admin
```

Options

Table B-9 Options and Arguments—vcb uvne delete

Option ***Argument***	**Description**
sysoid *sysoid*	The sysObject ID of the U-VNE configuration that you want to delete. **Note** Deleting the U-VNE does not delete or otherwise affect the U-VNE template from which the U-VNE was created.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-10 Error Codes—vcb uvne delete

Code	**Description**
112	The sysOID does not exist or already exists and is already modeled as a VNE.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

VCB CLI Command Reference: Device Types Commands

These topics provide reference information for the **vcb devicetype** commands you can use to create, view, modify, and delete user-experience attributes for device types:

- vcb devicetype add, page B-12
- vcb devicetype view, page B-14
- vcb devicetype modify, page B-15
- vcb devicetype delete, page B-16

vcb devicetype add

The **vcb devicetype add** command creates new user-experience attributes for the specified device type. Each device type is associated with a user-friendly name, icon, and device grouping.

Syntax

```
vcb devicetype add -devicetype device type -category prime network device category -name
device name [-key device type key]-user root -password admin
```

Description

The **vcb devicetype add** command creates user-experience attributes that affect how a device type is managed and displayed in Prime Network.

Note This command is typically used before adding a new U-VNE using the **vcb uvne add** command. For more information, see vcb uvne add, page B-4.

Usage Example

```
vcb devicetype add -devicetype CISCO_1760 -category ROUTER -name "Cisco 1760 Router"
-user root -password admin
```

Adds a device-type definition for the Cisco 1760 router, including the device category and user-friendly name that will appear in the Prime Network UI. Reference this definition when adding U-VNE definitions for a Cisco 1760 device using the **vcb uvne add** command.

Options

Table B-11 Options and Arguments—vcb devicetype add

Option *Argument*	Description
devicetype *device type*	The name to assign to the new device type. Each name must be unique. To see existing device types, use the **vcb devicetype view** command.
category *prime network device category*	The category to assign to the new device type, entered as a string (router, switch, unknown, and so on). By default, the device category defined in the U-VNE template is used.
name *device name*	The name to display for this device type in Prime Network. By default, the name is empty. **Note** The name need not be unique. The VCB does not enforce a naming convention for this value.
key *device type key*	(Optional) The unique ID of the new device type. This value is not displayed in Prime Network. **Note** We recommend that you *not* use this option, and let the VCB define the key instead.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-12 Error Codes—vcb devicetype add

Code	Description
501	Device type name already exists in the deviceTypes
502	Category value does not match any of the possible values
503	Key for this template is not unique
504	Device type name contains illegal characters
505	SysOID not found or not bound to device type in devicetypes.xml

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb devicetype view

The **vcb devicetype view** command returns an existing device-type association.

Syntax

```
vcb devicetype view -devicetype {device type | all} -user root -password admin
```

Description

Use the **vcb devicetype view** command to:

- Display user-experience attributes based on the specified device type.
- Display the list of device types defined in the system.
- Display all available device categories.

Usage Examples

Example 1

```
vcb devicetype view -devicetype all -user root -password admin
```

Returns a list of all device types defined in Prime Network.

Example 2

```
vcb devicetype view -devicetype CISCO_1760 -user root -password admin
```

Returns the device type details for device type CISCO_1760, including the category and user-friendly name defined for this type.

Options

Table B-13 Options and Arguments—vcb devicetype view

Option *Argument*	Description
devicetype *device type*	The name of the device type user-experience attributes (including name and device category) that you want to view. **Tip** Enter **all** as the *devicetype* to view a list of all device types defined in the system.
category all	Returns all defined device categories (router, switch, and so on) and their numeric equivalents.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-14 Error Codes—vcb devicetype view

Code	Description
511	deviceType not found

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb devicetype modify

The **vcb devicetype modify** command modifies the user-experience attributes associated with specific device types.

Syntax

```
vcb devicetype modify -devicetype device type name [-category prime network device
category] [-name device name] [-key device type key]-user username -password password
```

Description

The **vcb devicetype modify** command overwrites the user-experience settings defined for a device type, including the name, icon, and device category as they appear in Prime Network.

Usage Example

```
vcb devicetype modify -devicetype CISCO_1760 -category DSLAM -name "Cisco 1760 DSLAM"
-user root -password admin
```

Modifies the category and name assigned to the specified device type. Any VNE that uses the U-VNE-driver associated with this device type inherits these modified user-experience attributes.

Options

Table B-15 Options and Arguments—vcb devicetype modify

Option *Argument*	Description
devicetype *device type*	The name of the device type user-experience attributes that you want to modify.
category *prime network device category*	(Optional) Modifies the category assigned to the device type. Enter the category as a string (router, switch, unknown, and so on).
name *device name*	(Optional) Modifies the name that is displayed for this device type in Prime Network.
key *device type key*	(Optional) Modifies the unique ID of the device type. This value is not displayed in Prime Network. **Note** We recommend that you *not* use this option, and let the VCB define the key instead.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-16 Error Codes—vcb devicetype modify

Code	Description
502	Category value does not match any of the possible values for this enum
503	Key for this device type is not unique
504	Device type name contains illegal characters
511	Device type not found

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb devicetype delete

The **vcb devicetype delete** command deletes the user-experience attributes that are defined for the specified device type.

Syntax

```
vcb devicetype delete -devicetype device type -userdefined -user username
-password password
```

Description

The **vcb devicetype delete** command deletes the user-friendly name, icon, and grouping that were defined for the specified device type from the site.xml file. It does not delete or otherwise modify the U-VNE template from which the U-VNE for this device type was created.

Usage Example

```
vcb devicetype delete -devicetype CISCO_1760 -userdefined -user root -password admin
```

Deletes the user-experience attributes defined for the CISCO_1760 device type.

Options

Table B-17 Options and Arguments—vcb devicetype delete

Option *Argument*	Description
devicetype *device type*	The name of the device type from which you want to delete user-experience attributes.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-18 Error Codes—vcb devicetype delete

Code	Description
511	Device type not found.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

VCB CLI Reference: Standard and Pluggable Modules Commands

You can use the VCB CLI to create and manage standard and pluggable modules as described in these topics:

- VCB CLI Command Reference: Standard Modules, page B-18
- VCB CLI Command Reference: Pluggable Modules, page B-25

VCB CLI Command Reference: Standard Modules

These topics provide reference information for the **vcb module** commands you can use to create, view, and delete standard modules:

- vcb module add, page B-18
- vcb module view, page B-20
- vcb module delete, page B-24

vcb module add

The **vcb module add** command adds support for a new module type by creating a new registration from the specified template. This support enables VNEs that contain this module to properly recognize and model it in Prime Network.

Syntax

```
vcb module add -module module identifier -template template name -group module group
[ -hardwaredesc Hardware Description of the Module ] -user username -password password
```

```
vcb module add -module module identifier -template template name -sysoid <sysobject ID
devicefamily under which module should be supported> -scheme <ipcore|product> [
-hardwaredesc Hardware Description of the Module ] -user username -password password
```

Description

The **vcb module add** command enables the VNE to automatically detect a physical module based on an implementation defined in a module template. It can be used to:

- Create a module definition based on the module identifier. When this option is used, any VNE that uses the same spec file for its module definitions can model the new module.
- Create a module definition based on an extension to the definition used by the driver of a specific device (defined by its sysOID). When this option is used, only this particular device can model the new module.

This command updates the site.xml file, in which customizations are stored. Doing so enables the tool to differentiate between factory defaults (changes supplied from DSPs) and changes initiated by the VCB.

Usage Examples

Example 1

```
vcb module add -module ".1.3.6.1.4.1.9.12.3.1.9.29.99" -template "1.2.3.4" -group
ciscoentitymibspec -hardwaredesc cevCat6kWsf6kpfc3b -user root -password admin
```

Adds support for a module with the module ID .1.3.6.1.4.1.9.12.3.1.9.29.99, based on the 1.2.3.4 module template, which is listed in the spec group, ciscoentitymibspec. The hardware description is cevCat6kWsf6kpfc3b.

Example 2

```
vcb module add -module ".1.3.6.1.4.1.9.12.3.1.9.29.99" -sysoid "7.7.7"
-template "1.2.3.4" -scheme product -user root -password admin
```

Adds support for a module with the ID .1.3.6.1.4.1.9.12.3.1.9.29.99, based on module template 1.2.3.4. The VCB looks up the module spec used by the VNE-driver with the sysOID 7.7.7, and registers the new module for this device only (not for all devices that share the same module spec file).

Options

Table B-19 Options and Arguments—vcb module add

Option ***Argument***	**Description**
module *module identifier*	The module identifier, a number, string, or OID. The module identifier should be unique and new. **Note** The value specified for this argument must match the value returned from the physical command investigation.
group *module group*	The module group is a repository that might be shared across multiple device types. It specifies where the template is located. For the module groups that you can extend, see Module Groups and Module Specification Files, page B-66. **Note** Regular modules and pluggable modules use different module spec files. For information about pluggable modules, see VCB CLI Command Reference: Pluggable Modules, page B-25.
sysoid *sysoid*	The sysObjectID of the device that belongs to particular device family for which a concrete module is created using the implementation defined in the template. **Note** This sysObjectID must already exist in the system.
scheme *scheme*	Scheme name which will be used while adding the VNE.
template *template name*	The name of the template that contains the implementation of the physical investigation of this module. **Tip** The template option need not be specified if the module does not support ports; SIP and other carrier cards are examples of modules that do not support ports.
hardwaredesc *hardware description*	(Optional) Hardware description of the given module.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-20 Error Codes—vcb module add

Code	Description
103	No such template name in the templates file.
111	The SysOID specified does not exist in site.xml.
401	Module already exists in the spec file or in site.xml (thus is already modeled).
402	Module template file not found.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb module view

The **vcb module view** command returns an existing module configuration.

Syntax

```
vcb module view -sysoid sysoid -module {module identifier | all}-user root -password admin
-scheme <scheme>
```

```
vcb module view -group module group -module {module identifier | all}-user root -password
admin
```

```
vcb module view -group module group -template {template name | all}-user root -password
admin
```

Description

Use the **vcb module view** command to:

- Display module information based on the driver associated with the defined sysOID.
- Display details about the specified module.
- Display available templates found in the specified module spec file.

Usage Examples

Example 1

```
vcb module view -sysoid .1.2.3.4 -module all -user root -password admin -scheme product
```

Returns the list of supported modules for the device with sysOID 1.2.3.4.

Example 2

```
vcb module view -sysoid .1.2.3.4 -module 50068 -user root -password admin -scheme product
```

Returns the port layer definitions for module 50068, which was added to the VNE-driver for the device with sysOID 1.2.3.4.

Example 3

```
vcb module view -group ciscophysicalspec2 -module all -user root -password admin
```

Returns a list of all supported modules contained in the specified module group.

Example 4

```
vcb module view -group ciscophysicalspec2 -module 20091 -user root -password admin
```

Returns the port layer definitions for module 20091, which is part of the group ciscophysicalspec2.

Example 5

```
vcb module view -group ciscoentitymibspec -template all -user root -password admin
```

Returns a list of all templates defined in the specified module group.

Example 6

```
vcb module view -group ciscoentitymibspec -template ethernetDefault -user root -password
admin
```

Returns the port layer definitions of the template if it is defined in the specified module group.

Options

Table B-21 Options and Arguments—vcb module view

Option ***Argument***	**Description**
sysoid *sysoid*	The sysOID of the device that contains the module configuration that you want to view.
module *module identifier*	The module whose port layer configuration you want to view. The identifier can be a number, string, or OID, depending on the type of identifier used by the relevant device type. **Tip** Enter **all** as the *module identifier* to return a list of all modules in the defined device or group, listed by identifier.
group *module group*	The module spec group associated with the module or template whose details you want to view.
template *template name*	The name of the module template whose port layer configuration you want to view. **Tip** Enter **all** as the template name to return the list of all module templates contained in the specified module spec group.
- scheme *scheme*	Scheme name which will be used while adding the VNE.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-22 Error Codes—vcb module view

Code	Description
103	No such template name in the template file
402	Module template file not found
411	SysOID specified does not exist in site.xml or vendor file
412	Module identifier specified does not exist

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb module modify

Use the **vcb module modify** command to modify the associated module template.

Syntax

```
vcb module modify -module module identifier -group module group -template template name
[ -hardwaredesc Hardware Description of the Module -user username -password password
```

```
vcb module modify -module module identifier -sysoid device sysoid -scheme
<ipcore|product> -template template name [ -hardwaredesc Hardware Description of the
Module ] -user username -password password
```

Description

The **vcb module modify** command can be used to:

- Associate a module with another module template.
- Change the association between a module and its module template for a device specified by its sysOID.

Usage Examples

Example 1

```
vcb module modify -module 20091 -group ciscophysicalspec2
-template GE-fiberoptic-ethernet-default -user root -password admin
```

Modifies the registration for module 20091 by associating it with module template GE-fiberoptic-ethernet-default, which is part of group ciscophysicalspec2. This module definition replaces the one obtained from the template that was specified when the module was first added. This change affects all devices that contain this module.

Example 2

```
vcb module modify -sysoid .1.2.3.4 -module 50068 -group ciscophysicalspec2
-template oc3-ppp-default -user root -password admin -scheme product
```

Modifies the registration for module 50068 by associating it with module template cc3-ppp-default. This change affects only the device with sysOID 1.2.3.4.

Options

Table B-23 Options and Arguments—vcb module modify

Option *Argument*	Description
module *module identifier*	The module configuration that you want to modify. The identifier can be a number, string, or OID, depending on the type of identifier used by the relevant device type. **Note** The value specified for this argument must match the value returned by the device when investigating this module.
group *module group*	The name of the group that contains the module template to apply to the module.
sysoid *sysoid*	The sysObject ID of the device that contains the module configuration that you want to modify. **Note** You must specify a sysoid that was added using the VCB.
template *template name*	The name of the module template with which the defined module should be associated. Use this option to change the module template from which the module derives its configuration.
hardwaredesc *hardware description*	(Optional) Hardware description of the given module.
scheme *scheme*	Scheme name which will be used while adding the VNE.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-24 Error Codes—vcb module modify

Code	Description
103	No such template name in the template file
402	Module template file not found
411	SysOID specified does not exist in site.xml or vendor file
412	Module identifier specified not found or already exists in vendor spec file

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb module delete

Use the **vcb module delete** command to delete the association between a module and the implementation defined in a module template.

Syntax

vcb module delete -module *module identifier* **-group** *module group* **-user** *username* **-password** *password*

vcb module delete -module *module identifier* **-sysoid** *device sysoid* **-scheme** *<ipcore|product>* **-user** *username* **-password** *password*

Description

Note Use the **vcb module delete** command to enable VNEs to use factory-supplied updates if they become available.

The **vcb module delete** command is useful when migrating from an interim solution to a more complete implementation. Removing the association to the module from the site.xml file enables Prime Network to automatically detect a deployed implementation in a DSP.

Use the **vcb module delete** command:

- To delete the association between a module and its implementation on a specific device based on a defined sysOID. Other VNEs that use this implementation will still be able to identify the module.
- To delete the association between a module and its implementation on all devices that use the module as specified by the relevant group name. When the **-group** option is used, any VNEs that used this implementation will no longer be able to identify the module.

Note

- You can delete only those modules that were originally added using the VCB. Module definitions added as part of a developed VNE cannot be deleted.
- You can delete the module association on a specific device only when the association was created for that specific device.
- Deleting the module configuration does not delete or otherwise affect the template from which the configuration was created.

Usage Examples

Example 1

vcb module delete -module *20091* **-group** *ciscophysicalspec2* **-user** *root* **-password** *admin*

Deletes the definition for module 20091 from the group ciscophysicalspec2. The module definition is deleted from all devices that contain this module.

Example 2

vcb module delete -sysoid *.1.2.3.4* **-module** *50068* **-user** *root* **-password** *admin* **-*scheme*** *product*

Deletes the definition for module 50068 from the device with sysOID 1.2.3.4. Other devices that contain this module are unaffected.

Options

Table B-25 Options and Arguments—vcb module delete

Option *Argument*	Description
module *module identifier*	The module configuration that you want to delete. The identifier can be a number, string, or OID, depending on the type of identifier used by the relevant device type.
sysoid *device sysoid*	The sysObject ID of the device that contains the module configuration that you want to delete.
group *module group*	The module spec group that contains the template whose implementation you want to delete from the module.
scheme *scheme*	Scheme name which will be used while adding the VNE.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-26 Error Codes—vcb module delete

Code	Description
402	Module template file not found.
411	sysOID does not exist in the site.xml file or vendor spec file.
412	Module identifier not found or already exists in the vendor spec file.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

VCB CLI Command Reference: Pluggable Modules

These topics provide reference information for the **vcb pluggablemodule** commands you can use to create, view, modify, and delete pluggable modules:

- vcb pluggablemodule add, page B-26
- vcb pluggablemodule view, page B-28
- vcb pluggablemodule modify, page B-29
- vcb pluggablemodule delete, page B-30

Note To add, view, modify, or delete regular modules, see VCB CLI Command Reference: Standard Modules, page B-18.

vcb pluggablemodule add

Use the **vcb pluggablemodule add** command to create a pluggable module definition that is associated using the module identifier. Use this command for pluggable modules such as SFPs and XFPs.

Syntax

```
vcb pluggablemodule add -group groupName -module moduleno -pid pid -mediatype MediaType
-pluggabletype pluggableType -user username -password password
```

```
vcb pluggablemodule add -group groupName -containeroid containeroid -user username
-password password
```

```
vcb pluggablemodule add -group groupName -basetype basetype -user username -password
password
```

Description

Prime Network does not model pluggable modules, only the ports on them. Configuration details that are entered using this command are displayed at the port level to differentiate between pluggable and regular ports. Changes made through this command apply to all VNEs that use the same spec file for pluggable ports.

The container oid and base type value are additional configuration information needed to model pluggable module properly. When you add new pluggable module make sure that container oid and base type values are added in the system.

Usage Example

```
vcb pluggablemodule add -module ".1.3.6.1.4.1.9.12.3.1.9.51.22"
–group pluggable-ports-spec –mediatype fiber –pluggabletype SFP -user root -password
admin
```

Adds support for a pluggable module of type SFP with the module ID .1.3.6.1.4.1.9.12.3.1.9.51.22, to the pluggable-ports-spec module specification file. The media type is fiber optic.

```
vcb pluggablemodule add –group pluggable-ports-spec –basetype .1.3.6.1.4.1.9.12.3.1.9.52
-user root -password admin
```

Add pluggable module base type configuration with the base type value .1.3.6.1.4.1.9.12.3.1.9.52, to the pluggable-ports-spec module specification file.

```
vcb pluggablemodule add –group pluggable-ports-spec -containeroid
.1.3.6.1.4.1.9.12.3.1.5.153  -user root -password admin
```

Add pluggable module container oid configuration with the container oid .1.3.6.1.4.1.9.12.3.1.9.52, to the pluggable-ports-spec module specification file.

Options

Table B-27 Options and Arguments—vcb pluggablemodule add

Option *Argument*	Description
module *moduleno*	The identifier for the module that you want to add. The identifier can be a number, string, or OID, depending on the type of identifier used by the relevant device type. **Note** The value specified for this argument must match the value returned by the device when investigating this module.
group *pluggable port group*	The name of the group.
pid *pid*	The pid for the module.
mediatype *MediaType*	The media type of the port on the pluggable module.
pluggable type *pluggableType*	The type of the pluggable module; one of SFP, XFP, X2, XENPAK.
containeroid	The oid of a pluggable module container where the pluggable module will be inserted.
basetype	The oid of a pluggable module without the last octet in the oid For example, if the oid of a pluggable module is .1.3.6.1.4.1.9.12.3.1.9.52.10 then the basetype oid would be .1.3.6.1.4.1.9.12.3.1.9.52

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-28 Error Codes—vcb pluggablemodule add

Code	Description
403	Pluggable module spec file not found
404	Invalid pluggable module type

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb pluggablemodule view

Use the **vcb pluggablemodule view** command to display details for a pluggable module.

Syntax

```
vcb pluggablemodule view -group groupName -module moduleno | all [-userdefined]-user
username -password password
```

```
vcb pluggablemodule view -group groupName -containeroid oid | all [-userdefined]-user
username -password password
```

```
vcb pluggablemodule view -group groupName -basetype basetype| all [-userdefined]-user
username -password password
```

Description

Use the **vcb pluggablemodule view** command to display the pluggable module details—such as pid, media type, and pluggable type—for one or all modules in the given pluggable port group.

Usage Examples

```
vcb pluggablemodule view -group pluggable-port-spec -module
".1.3.6.1.4.1.9.12.3.1.9.51.22" -user root -password admin
```

Returns the pluggable module details—such as pid, media type, and pluggable type—for the module with the given vendor oid.

```
vcb pluggablemodule view -group pluggable-port-spec -containeroid all -user root
-password admin
```

Returns the pluggable module container oid details, if the list has userdefined container oid, it is marked with # after the oid.

```
vcb pluggablemodule view -group pluggable-port-spec -basetype all -user root -password
admin
```

Returns the pluggable module details.

Options

Table B-29 Options and Arguments—vcb pluggablemodule view

Option *Argument*	Description
module *moduleno*	The identifier for the module that you want to view. The identifier can be a number, string, or OID, depending on the type of identifier used by the relevant device type. **Tip** Enter **all** as the *module no* to return the list of all modules contained in the specified pluggable port group.
group *groupname*	The name of the group.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-30 Error Codes—vcb pluggablemodule view

Code	Description
403	Pluggable module spec file not found.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb pluggablemodule modify

Use the **vcb pluggablemodule modify** command to change the pluggable module configuration.

Syntax

```
vcb pluggablemodule modify -group groupName -module moduleno -pid pid {[-mediatype
MediaType] | [-pluggabletype pluggabletype]}
```

Description

The **vcb pluggablemodule modify** command enables you to modify the configuration for a pluggable module.

Usage Example

```
vcb module modify -module ".1.3.6.1.4.1.9.12.3.1.9.51.22" -group pluggable-ports-spec
-mediatype "fiber optic" -user root -password admin
```

Modifies the mediaType attribute of the pluggable module with identifier .1.3.6.1.4.1.9.12.3.1.9.51.22.

Options

Table B-31 Options and Arguments—vcb pluggablemodule modify

Option *Argument*	Description
module *moduleno*	The identifier for the pluggable module configuration that you want to modify. The identifier can be a number, string, or OID, depending on the type of identifier used by the relevant device type.
group *pluggable port group*	The name of the group.
pid *pid*	The pid for the module.
mediatype *MediaType*	(Optional) The media type of the port on the pluggable module.
pluggable type *pluggableType*	(Optional) The type of the pluggable module; one of SFP, XFP, X2, AND XENPAK.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-32 Error Codes—vcb pluggablemodule modify

Code	Description
403	Pluggable module spec file not found
404	Invalid pluggable module type

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb pluggablemodule delete

Use the **vcb pluggablemodule delete** command to delete a pluggable module that was previously added using the VCB.

Syntax

```
vcb pluggablemodule delete -group pluggableportgroupN -module moduleno -user username
-password password
```

```
vcb pluggablemodule delete -group pluggableportgroup -containeroid container oid -user
username -password password
```

```
vcb pluggablemodule delete -group pluggableportgroup -basetype basetype oid -user
username -password password
```

Description

Use the **vcb pluggablemodule delete** command to delete pluggable modules that were created using the VCB.

Note Use the **vcb pluggablemodule delete** command to enable VNEs to use factory-supplied updates when they are available.

Usage Examples

```
vcb pluggablemodule delete -module ".1.3.6.1.4.1.9.12.3.1.9.51.22" -group
pluggable-ports-spec -user root -password admin
```

Removes the definition for the pluggable module with identifier .1.3.6.1.4.1.9.12.3.1.9.51.22.

```
vcb pluggablemodule delete -basetype ".1.3.6.1.4.1.9.12.3.1.9.52" -group
pluggable-ports-spec -user root -password admin
```

Removes the definition for the pluggable module basetype with identifier .1.3.6.1.4.1.9.12.3.1.9.52.

```
vcb pluggablemodule delete -containeroid ".1.3.6.1.4.1.9.12.3.1.5.153" -group
pluggable-ports-spec -user root -password admin
```

Removes the definition for the pluggable module container oid with identifier .1.3.6.1.4.1.9.12.3.1.5.153.

Options

Table B-33 Options and Arguments—vcb pluggablemodule delete

Option *Argument*	Description
module *moduleno*	The identifier for the pluggable module that you want to delete. The identifier can be a number, string, or OID, depending on the type of identifier used by the relevant device type.
group *pluggable port group*	The name of the group from which to delete the pluggable module.

Note For the list of global options, see Global Command Options, page B-2.

VCB CLI Reference: Events Commands

This section describes the CLI commands that can be used to customize events, as follows:

- VCB CLI Reference: vcb event Commands, page B-31
- VCB CLI Reference: vcb eventparsingrules Commands, page B-40
- VCB CLI Reference: vcb eventpattern Commands, page B-47
- VCB CLI Reference: vcb eventarg Command, page B-53

VCB CLI Reference: vcb event Commands

These topics provide reference information for the **vcb event** commands you can use to create, view, modify, and delete events:

- vcb event add, page B-31
- vcb event view, page B-34
- vcb event modify, page B-37
- vcb event delete, page B-39

vcb event add

Use the **vcb event add** command to create an event definition for a syslog or a trap in Prime Network based on user input.

Note To create a script to add unsupported traps from a MIB, use the e **vcb event view** command with the **-generatecli** option.To list the traps in a MIB that are supported and those that are not, use the **vcb event view** command with the **-mibfile** option. For more information, see vcb event view, page B-34.

Syntax

```
vcb event add -eventtype {syslog|trap} -eventname eventName [-alarmid alarmId]
{-subtype1 subtype1Name [-ticketable1]
[-severity1 critical|major|minor|warning|info|cleared>]
[-shortdesc1 short description string] [-autoclear1 false|true]
. . .
[-subtypen subtypenName] [-ticketablen][-severityn
critical|major|minor|warning|info|cleared ]
[ -shortdescn short description string] [-autoclearn false|true]

-user username -password password
```

Description

The **vcb event add** command creates an event definition in Prime Network based on the user input. Afterwards, the VNE-driver can create specific instances of this event for incoming traps or syslogs, persist them in the event database, and forward them to interested clients.

Usage Example

```
vcb event add -eventtype syslog
-eventname "stack switch status syslog"
-subtype1 "stack switch removed syslog"
-severity1 minor
-subtype2 "stack switch added syslog"
-severity2 cleared
-user root -password admin
-faultnature ADAC
-faultcategory "QOS"
```

Adds a syslog event definition with the name “stack switch status syslog”. Two subtypes are added: “stack switch removed syslog” with severity minor and “stack switch added syslog” with severity cleared. By default, the subevents are not ticketable.

The syslog for which the event definition was added is:

```
STACKMGR-4-SWITCH_[ADDED|REMOVED]: Switch [dec] has been [ADDED to|REMOVED from] the
stack.
```

A device sends one syslog when a switch is added to a stacked device (clear alarm) and another syslog when a switch is removed from a stacked device (asserted minor alarm).

Options

Table B-34 Options and Arguments—vcb event add

Option *Argument*	Description
eventtype *event type*	Type of event. Valid values are: • syslog • trap
eventname *event name*	Unique string that identifies the event within Prime Network.

Table B-34 Options and Arguments—vcb event add (continued)

Option *Argument*	**Description**
alarmid *alarm ID*	(Optional) Unique integer identifier for the event. **Note** Recommendation—Do not provide this argument; the VCB automatically generates a unique number.
subtype$_n$ *subtypen name*	Unique string that identifies the subevent with the Prime Network event.
ticketable$_n$	(Optional) Optional parameter for subtype. If specified, indicates that a ticket should be generated for this subevent. By default, no ticket is generated for the subtype.
-autoclear$_n$ *`false`*|*`true`*	(Optional) Optional parameter for subtype. If the event is ticketable, setting autoclear to false causes the subevent to remain asserted until the clear alarm arrives or the user manually acknowledges or clears the subevent. **Note** Root cause events are not autocleared even when autoclear is set to false. By default, autoclear is true for user-defined event definitions.
severity$_n$ *severity level*	Optional parameter for subtype. The severity of the subevent. Possible values are critical, major, minor, warning, info, and cleared. Info is the default severity value for a subevent.
shortdesc$_n$ *short description*	Optional parameter for subtype. A short description of the subevent. This string is stored in the event database. By default, the subtype value is used as the default shortdesc value.
-faultnature	Parameter that indicates how the fault is cleared, either manually or automatically. Possible values are: • ADAC (Automatically Detected Automatically Cleared) - The event is automatically detected and automatically cleared by the system. For example, “link down” event. • ADMC (Automatically Detected Manually Cleared - The event must be manually cleared by the user. For example, “DWDM fatal error” syslog.

Table B-34 Options and Arguments—vcb event add (continued)

Option *Argument*	Description
-faultcategory	Event category (3GPP standards). Possible values are: • COMMUNICATIONS • QOS • PROCESSING • EQUIPMENT • ENVIRONMENTAL • UNDETERMINED • INTEGRITYVIOLATION • OPERATIONALVIOLATION • PHYSICALVIOLATION • SECURITYORSERVICEMECHANISMVIOLATION • TIMEDOMAINVIOLATION

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-35 Error Codes—vcb event add

Code	Description
201	Event name already exists in Prime Network.
202	Alarm ID already exists in Prime Network.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb event view

Use the **vcb event view** command to list event definition registrations.

Syntax

```
vcb event view -eventname {eventName | all} [-substringmatch]} -user username
-password password -eventtype {trap | syslog | service}
```

```
vcb event view -genericevents all | trap | syslog [-ipaddress vneip] [-date yyyy-mm-dd]
[-time hh:mm:ss] [-maxrecords num]-user username -password password
```

```
vcb event view -mibfile complete-path-mibFilename [-generatecli -repository
ParsingrulesHive - group PatternsHive] -user username -password password
```

Description

The **vcb event view** command enables you to view event definitions including event properties such as alarm ID, event subtypes, severity, and ticketability.

Usage Examples

```
vcb event view –userdefined –eventname all -user root -password admin
```

Returns all the event definitions that were added to Prime Network using the VCB.

```
vcb event view –eventname bgp –substringmatch -user root -password admin
```

Returns all BGP event definitions in Prime Network, including those that were added using the VCB.

```
vcb event view -user root -password admin -eventname bgp -substringmatch -eventtype
service
```

Returns all BGP service events.

```
vcb event view -mibfile /mibs/IF-MIB –user root -password admin
```

Returns lists of supported events and unsupported events based on the traps in the IF-MIB file.

```
vcb event view -mibfile /mibs/IF-MIB -generateeventcli
-group cisco-trap-product-parsing-rules -repository cisco-trap-repository –user root
–password admin
```

Creates, but does not run, a script /Main/VcbEventCommand.sh. The script contains three **vcb** commands for each unsupported trap; the commands add an event (and provide an event ID), event parsing rules, and an event pattern. Optionally, edit the script.

To run the script, change permissions on the file to ensure that it is executable and supply a username and password as input; see this example:

```
chmod 755 VcbEventCommand.sh
./VcbEventCommand.sh -user root -password admin
```

Note The Prime Network gateway maintains a known list of MIBs that are used to provide translation for trap varbinds when displayed in the UI. When an event is added from a MIB that is unknown to the gateway, the VCB does not add the MIB to the known MIB list. As a result, the varbinds for this trap might not be translated to user-friendly names.

Options

Table B-36 Options and Arguments—vcb event view

Option *Argument*	**Description**
eventname *eventName*	Unique string that represents the event. **Tip** Enter **all** as the *eventName* to display information on all the event definitions in Prime Network. Use this argument with caution because the number of events can potentially be very large.
substringmatch	(Optional) Indicates that the event name argument is not an exact match.
eventtype *trap* | *syslog* | *service*	Displays events of a specific type - traps, syslogs or service events.

Table B-36 Options and Arguments—vcb event view (continued)

Option *Argument*	Description
genericevents *generic event type*	Displays all events from the Prime Network database (in raw format). **Tip** Enter **all** as the *generic event type* to display information on all the events in Prime Network.
ipaddress *neip*	(Optional) Generic events filter. IP address for the NE for which you want to see generic events.
date *yyyy-mm-dd*	(Optional) Generic events filter. The date after which the events arrived.(Returns events that arrived after the given day.)
time *hh:mm:ss*	(Optional) Generic events filter. The time after which the events arrived. (Returns events that arrived after the given time.)
maxrecords *num*	(Optional) Generic events filter. The maximum number of events that you want to display. The default value is 100.
mibfile `complete-path-mibFilename`	Loads MIB modules and compares the traps defined in the MIB against the events that are supported in Prime Network. Displays lists of supported traps and unsupported traps. **Note** Before using this command option, copy the MIB and dependent MIB files to a local folder. Rename each MIB file, removing the .my file extension from it.
generatecli	(Optional) When provided, produces a script, *NETWORKHOME*/Main/VcbEventCommand.sh. The script contains commands to add basic event support for each unsupported trap that was identified through the **-mibfile** option.
repository *ParsingrulesHive*	Mandatory **-generatecli** option. Hive that includes event parsing rules for traps.
group *PatternsHive*	Mandatory **-generatecli** option. Hive that includes event patterns for traps.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-37 Error Codes—vcb event view

Code	Description
231	No such event exists in the events file

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb event modify

Use the **vcb event modify** command to modify events that were previously defined using the VCB or to modify event attributes for factory-defined events (by using the **-override** option). This command can also be used to drop events.

Syntax

```
vcb event modify -eventname eventName [-alarmid alarmId] [-override]
{-subtype1 subtype1Name {[-ticketable1] [-autoclear1 false|true]-severity1
critical|major|minor|warning|info|cleared
-shortdesc1 short description string…
{-subtypen subtypenName [-ticketablen] [-autoclear1 false|true]
-severityn critical|major|minor|warning|info|cleared
[-shortdescn] short description string -user username -password password
[-eventtype {trap | syslog | service | drop}
```

Description

The **vcb event modify** command modifies an event definition in Prime Network. It can also be used to instruct the system to drop a specific event.

Note Support for modifying an event is limited due to the complexity involved. When additional changes are required—such as changing the name of an event or a subtype—the supported procedure is to delete the entire event definition and add it afresh:

- Delete the event, event pattern, and associated event parsing rules.
- Add the event, event pattern, and event parsing rules.

Usage Examples

```
vcb event modify -eventname "stack switch status syslog"
```

```
-subtype1 "stack switch removed syslog" -severity1 major -ticketable1 -user root
-password admin
```

Updates the event definition for the stack switch status syslog, changing the severity of the specified subtype to major and making the subtype ticketable. For a corresponding example of how this event was added, see Usage Example, page B-32 for the **vcb event add**.

```
vcb event modify -eventname "bgp trap" -override -eventtype drop -user root -password admin
```

Drops the “bgp trap” event. This overrides the system-defined event.

Options

Table B-38 Options and Arguments—vcb event modify

Option *Argument*	Description
eventname *eventName*	Unique string identifies the event within Prime Network.
eventtype *trap* I *syslog* I *service* I *drop*	Modifies an event of a specific type or instructs the system to drop an event.

Table B-38 Options and Arguments—vcb event modify (continued)

Option ***Argument***	**Description**
alarmid *alarmId*	(Optional) Unique integer identifier for the event. If not provided, the VCB automatically generates a unique number. **Note** We recommend that you do not provide an input alarm ID.
subtype*n* *subtypenName*	Unique string that identifies the subevent with the Prime Network event. To retain ticketability for any ticketable subtype—whether you want to modify the subtype or not —you must enter the subtype option and argument along with the ticketable option (below).
ticketable*n*	(Optional) Parameter for subtype. Indicates whether a ticket should be generated for this subtype. If not specified, no ticket is generated. **Note** To retain ticketability, supply the ticketable option for all subtypes that are currently defined as ticketable events (even for subtypes that you do not intend to modify). Otherwise, the subtypes are modified to be non-ticketable events.
-autoclear$_n$ `false`|`true`	(Optional) Parameter for subtype. If the event is ticketable, setting autoclear to false causes the subevent to remain asserted until the clear alarm arrives or the user manually acknowledges or clears the subevent. **Note** Root cause events are not autocleared even when autoclear is set to false. By default, autoclear is true for user-defined event definitions.
severity*n value*	(Optional) Parameter for subtype. Specifies the severity of the subevent. Possible values are critical, major, minor, warning, info, and cleared.
shortdesc*n* `short description string`	(Optional) Parameter for subtype. A short description.
override	(Optional) Indicates that you expect to override attributes for a factory-defined event.
-faultnature	Parameter that indicates how the fault is cleared, either manually or automatically. Possible values are: • ADAC (Automatically Detected Automatically Cleared) - The event is automatically detected and automatically cleared by the system. For example, “link down” event. • ADMC (Automatically Detected Manually Cleared - The event must be manually cleared by the user. For example, “DWDM fatal error” syslog.

Table B-38 Options and Arguments—vcb event modify (continued)

Option *Argument*	Description
-faultcategory	Event category (3GPP standards). Possible values are: • COMMUNICATIONS • QOS • PROCESSING • EQUIPMENT • ENVIRONMENTAL • UNDETERMINED • INTEGRITYVIOLATION • OPERATIONALVIOLATION • PHYSICALVIOLATION • SECURITYORSERVICEMECHANISMVIOLATION • TIMEDOMAINVIOLATION

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-39 Error Codes—vcb event modify

Code	Description
202	Alarm ID already exists in Prime Network.
251	Event name does not exist in Prime Network.
252	Event subtype name does not exist for the event.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb event delete

Use the **vcb event delete** command to delete an event definition. The **vcb event delete** does not delete or change the event template from which the event definition was cloned.

Syntax

```
vcb event delete -eventname eventName -user username -password password
```

Description

The **vcb event delete** command removes event definitions created using the VCB and removes event attribute overrides from factory-defined events. Deleting a factory-defined event removes event attribute overrides only and not the event itself; the original event attributes are then applied to future events.

Usage Example

```
vcb event delete -eventname "stack switch status syslog" -user root -password admin
```

Deletes the event definition. All registry entries added as a part of the event add command are removed from the site.xml file.

Options

Table B-40 Options and Arguments—vcb event delete

Argument	Description
eventname *eventName*	Name of the event to be deleted

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-41 Error Codes—vcb event delete

Code	Description
231	No such event exists in the events file

Note For the list of general VCB error codes, see General Error Codes, page B-2.

VCB CLI Reference: vcb eventparsingrules Commands

These topics provide reference information for the **vcb eventparsingrules** commands you can use to create, view, modify, and delete event parsing rules:

- vcb eventparsingrules add, page B-41
- vcb eventparsingrules view, page B-43
- vcb eventparsingrules modify, page B-45
- vcb devicetype modify, page B-15
- vcb eventparsingrules delete, page B-46

vcb eventparsingrules add

Use the **vcb eventparsingrules add** command to create a VNE-driver registration for adding parsing rules to support a new trap or syslog by customizing a specified set of event templates.

Syntax

```
vcb eventparsingrules add -templates templateName1, templateName2, …, templateNamen
-group repository –rulename rulename [-enable]
{ [-arg1 -arg1Value…-argN argNValue] } -user username -password password
```

Description

The **vcb eventparsingrules add** command creates a VNE-driver event registration based on the templates chosen by the user. This enables Prime Network to identify and associate the event to a particular device component instead of classifying the event as a generic event.

The command does the following:

- Creates a separate registry configuration (a copy) for customizing the event. Parameters that you input using the command affect the copy.
- Creates rules for handling the event based on the event template and user input.
- Updates the site.xml file, so that Prime Network can differentiate customizations created using the VCB from changes supplied in VNE-driver registration files.

Usage Example

```
vcb eventparsingrules add -enable

   -templates
syslog-identification,syslog-subtype-from-expression,create-managedelement-key,create-a
na-syslog-event

   -group cisco-syslog-repository

   -rulename stack-switch-status-syslog

   -syslog_identification_testmessage "STACKMGR-4-SWITCH_ADDED: Switch 2 has been added
to the stack"

   -syslog_identification_expression "STACKMGR-4-SWITCH_%%subtypekey%%: Switch
%%uniqueid%% has been .*"

   -syslog_subtype_from_expression_replacing_rules "ADDED-stack switch added
syslog,REMOVED-stack switch removed syslog"

-create_ana_syslog_event_type "stack switch status syslog"

-user root -password admin
```

Adds parsing rules to identify the syslog correctly, associates it with the correct device component, and creates the corresponding Prime Network event and subevent.

Four event templates are entered:

- syslog-identification—Rules that pertain to syslog identification.
- syslog-subtype-from-expression—Rules that map the syslog values (in this case, ADDED and REMOVED) to the event subtype names: stack switch added syslog, stack switch removed syslog. The example includes two rules, separated by commas.

Note Rules must be comma-separated. Each rule must include a value and event subtype, separated by a hyphen: value-event subtype.

- create-managedelement-key—Indicates that the syslog should be associated with the managed element. (There are no input parameters for this template.)
- create-ana-syslog-event—Provides rules for creating instances of the corresponding Prime Network event (defined with the **vcb event add** command).

Input parameters for the event templates are variable arguments that depend on the templates selected.

- syslog_identification_expression—The actual syslog message with input that is of interest to the user and is masked with special keys, such as %%subtypekey%%, %%uniqueid%%, and %%entityid%%, depending on which is applicable. In the previous example, only subtypekey and uniqueid parameters are relevant.

Note Only a substring of the message is used in the example, because whatever comes afterward is of no interest to the user.

- syslog_subtype_from_expression_replacing_rules—Specifies the mapping from the subtypekey to the subevent name. The subevent string should exactly match one of the subevent names that was defined using the **vcb event add** command.
- create_ana_syslog_event_type—Specifies the event name.

Note This parameter should exactly match the event name defined using the **vcb event add** command.

Options

Table B-42 Options and Arguments—vcb eventparsingrules add

Option ***Argument***	**Description**
templates *template name1, ... template namen*	Comma-separated list of event template names. Event templates are divided into categories that correspond to the function they fulfill (identification, association, and so on.) Depending upon the trap or syslog that you are adding, select no more than one template from each template category.
group *repository*	Specifies the vendor-specific trap or syslog repository file under which the customizations should be made.
rule *rulename*	String that is used as a key name for event rule definition.
enable	(Optional) Indicates whether the rule is enabled or disabled. Only enabled rules are used to parse incoming traps and syslogs.
variable arguments	Arguments vary from one event template to another event template.
syslog_identification_testmessage	(Optional) Parameter valid for syslogs only. An example syslog message used to check the correctness of the regular expression that the VCB creates automatically based on user input.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-43 Error Codes—vcb eventparsingrules add

Code	Description
211	Event template file not found.
103	No such template name in template file.
212	Only one template can be selected from each template category.
213	Invalid expression for syslog.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb eventparsingrules view

The **vcb eventparsingrules view** command displays event registrations. Use it to verify that you successfully added an event parsing rule or to view parameters (to fill them in based on the example).

Syntax

```
vcb eventparsingrules view -group repository name -rulename { rulename | all } [-detail]
[-userdefined] -user username -password password
```

```
vcb eventparsingrules view -template templateName | all -inputparam -user username
-password password
```

Description

The **vcb eventparsingrules view** command shows configuration settings. Use it to list:

- Details of the events repository for a particular rulename or for all the events in the file. The information displayed includes the parsing rules and important parameters in each rule.
- Input parameters in the event template specified by -template option. If all is specified, the user input parameters of all the templates are displayed.

Usage Examples

Example 1

```
vcb eventparsingrules view -template syslog-identification -inputparam -user root
-password admin
```

Displays the syslog-identification event template definition, including a detailed description of the input parameters required when using the template to add event parsing rules.

Example 2

```
vcb eventparsingrules view -group cisco-syslog-repository -userdefined -rulename all
-user root -password admin
```

Displays all event parsing rules that were defined using the VCB under the cisco-syslog-repository hive.

Example 3

```
vcb eventparsingrules view -group cisco-syslog-repository
-rulename stack-switch-status-syslog -user root -password admin
```

Displays the event parsing rules for the event “stack-switch-status-syslog” which was created using the VCB.

Example 4

```
vcb eventparsingrules view -group cisco-syslog-repository -rulename all -user root
-password admin
```

Displays all the event parsing rules present in the hive cisco-syslog-repository, including those added using the VCB.

Options

Table B-44 Options and Arguments—vcb eventparsingrules view

Option *Argument*	Description
group *repository name*	The trap or syslog repository filename.
rulename *ruleName*	The unique string that is used to represent the event parsing rules. **Tip** Enter **all** as the *ruleName* to display information on all the rules in Prime Network.
detail	(Optional) Lists the entire rule contents including the parsing rule entry details.
template *templateName*	The event template name. **Tip** Enter **all** as the *templateName* to display information on all event templates in Prime Network.
inputparam	(Optional) Lists template definition entries that require user input when creating event parsing rules.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-45 Error Codes—vcb eventparsingrules view

Code	Description
103	No such template name in the templates file.
222	Parsing rules repository not found.
231	No such rule name in the site.xml.

vcb eventparsingrules modify

Use the **vcb eventparsingrules modify** to modify the parsing rule definitions. The most common use case for this command is to select one or more different templates because the certification of the customization failed.

Syntax

```
vcb eventparsingrules modify -templateS templateName1, templateName2, …, templateNamen
-group repository -rulename rulename [-enable] { [-arg1 -arg1Value...-argN argNValue] } -user
username -password password
```

Description

The **vcb eventparsingrules modify** command changes parsing rule definitions based on the templates chosen by the user. The command can also be used to add parsing rules that were inadvertently omitted when adding the parsing rule. For example, use the command to add the rules for extracting the uniqueid parameter.

Options

Table B-46 Options and Arguments—vcb eventparsingrules modify

Option *Argument*	Description
templates *template name1, ...* *template namen*	Comma-separated list of event template names. Event templates are divided into categories that correspond to the function they fulfill (identification, association, and so on.) Depending upon the trap or syslog that you are adding, select no more than one template from each template category.
group *repository*	The hive under which the customizations should be made. The hive is the vendor-specific trap or syslog repository file.
rulename *ruleName*	String that is used as a key name for event rule definition.
enable	(Optional) Indicates whether the rule should be enabled or disabled. Only enabled rules are used to parse incoming traps and syslogs.
variable arguments	Each event template can require different input and a different number of input parameters from none to more than one. See Event Templates Input Summary—Required and Optional Input, page B-83.

Note **vcb eventparsingrules modify** command is not supported in Prime Network. To modify the parsing rules use *repository* and *rule name* option. For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-47 Error Codes—vcb eventparsingrules modify

Code	Description
103	No such template name in the templates file.
211	Event template file not found.
212	Only one template can be selected from each template category.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb eventparsingrules delete

Use the **vcb eventparsingrules delete** command to delete the parsing rule definitions of an event. Doing so does not delete or change the event template from which that event definition was cloned.

Syntax

```
vcb eventparsingrules delete -group repository hive -rulename rulename -user username
-password password
```

Description

The **vcb eventparsingrules delete** command removes event parsing rule definitions created from an event template. It does not change or delete the event template itself.

Usage Example

```
vcb eventparsingrules delete -group cisco-syslog-repository
-rulename stack-new-master-syslog
```

This example deletes the stack-new-master-syslog rule from the cisco-syslog-repository hive.

Options

Table B-48 Options and Arguments—vcb eventparsingrules delete

Option *Argument*	Description
group *repository hive*	The hive from which to remove the event parsing rule.
rulename *ruleName*	The rule to delete.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-49 Error Codes—vcb eventparsingrules delete

Code	Description
222	Parsing rules repository not found.
241	No such rule name in the site.xml.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

VCB CLI Reference: vcb eventpattern Commands

These topics provide reference information for the **vcb eventpattern** commands you can use to create, view, modify, and delete event patterns:

- vcb eventpattern add, page B-47
- vcb eventpattern view, page B-49
- vcb eventpattern modify, page B-50
- vcb eventpattern delete, page B-52

vcb eventpattern add

Use the **vcb eventpattern add** command to create a VNE-driver registration that points from the parsing rules hive, which is scheme or VNE-specific, to the parsing rules defined in the repository file.

Note Only those events that have this pointer are deemed as supported events. Other events are deemed generic events despite having parsing rules and event definitions.

Syntax

```
vcb eventpattern add [-patternid patternId] -group parsing rules hive
-repository parsing rules repository hive -rulename rulename -user username
-password password
```

Description

The **vcb eventpattern add** command creates a pointer from the parsing-rules hive to the repository where the actual parsing rules are defined.

Usage Examples

```
vcb eventpattern add
-patternid 202 -group cisco-syslog-product-parsing-rules
-repository cisco-syslog-repository
-rulename stack-switch-status-syslog -user username -password password
```

Adds a pointer from the parsing rules file to the actual definitions in the parsing-rules hive with pattern ID 202. It points to the key (rule) named stack-switch-status-syslog in the cisco-syslog-repository file.

Options

Table B-50 Options and Arguments—vcb eventpattern add

Option *Argument*	Description
patternid *patternId*	(Optional) (Recommendation: do not provide.) Unique integer to identify the supported event to VNEs. If not provided, VCB generates this number automatically. **Note** Omitting this option and argument enables the VCB to ensure that the patternid is unique and that it does not overlap with other file definitions due to registry inheritance.
group *parsing rules hive*	The hive to which this pattern should be added. The parsing-rules hives are generally scheme-specific. Device type-specific definitions can also be made.
repository *parsing rules repository hive*	The trap or syslog repository where the actual parsing rules are defined. **Note** Enter the same hive that was specified when creating parsing rules registrations using the **vcb eventparsingrules add** command.
rulename *rulename*	String that is used as a key name for event rule definition. **Note** Enter exactly the same string as the one that was specified when creating parsing rules registrations using the **vcb eventparsingrules add** command.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-51 Error Codes—vcb event pattern add

Code	Description
221	Parsing rules hive not found.
222	Parsing rules repository not found.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb eventpattern view

Use the **vcb eventpattern view** command to display event registrations. It is useful when you need to verify successful completion of an add command or to help find a similar case for filling in parameters on other commands.

Syntax

```
vcb eventpattern view -group parsingrules hive -rulename { rulename | all }
[-substringmatch] [-full] -user username -password password
```

Description

The **vcb eventpattern view** command shows the actual set of events that are supported by a particular NE type or scheme. It displays the pattern ID and the repository file where the event parsing rules are defined. When the substringmatch option is used, only rules that contain a certain substring are displayed; use this option, for example, to obtain rules for a technology name such as MPLS.

Usage Examples

Example 1

```
vcb eventpattern view -group cisco-syslog-parsing-rules -rulename
stack-switch-status-syslog -user root -password admin
```

Displays the event pattern definition for the specified rulename; that is, the pattern ID, and the pattern is pointing to the parsing rules repository.

Example 2

```
vcb eventpattern view -group cisco-syslog-parsing-rules -rulename all -user root
-password admin
```

This example shows all the event pattern definitions in the specified hive.

Example 3

```
vcb eventpattern view -group cisco-syslog-parsing-rules -userdefined

-rulename bgp -substringmatch -full -user root -password admin
```

This example shows the entire event definition for all BGP events (including those defined using the VCB) defined in the cisco-syslog-parsing-rules hive. The following information is displayed:

- Pattern definitions—Parsing rules repository, pattern ID
- Parsing rules definitions—All rules in the definition that require user input, and the values set for these parameters
- Event definitions—Event attributes such as eventname, subevent names, ticketability, severity and so on

Options

Table B-52 Options and Arguments—vcb eventpattern view

Option ***Argument***	**Description**
group *parsingrules hive*	The parsing-rules filename used by the NE type or scheme.
rulename *rule name*	Unique string that represents the event parsing rules. **Tip** Enter **all** as the *rule name* to list all event parsing rules defined in the repository.
substringmatch	(Optional) Indicates that the rule name provided is not an exact match. This option is useful when you want to know the names of all rules that belong to a particular technology, such as BGP.
full	(Optional) Displays the entire details of the event, from the pattern definition and parsing rules to the event definition. Provides the complete picture of the how an event is supported in Prime Network. **Note** Avoid this option when using the “all” argument because it can result in a very large output.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-53 Error Codes—vcb eventpattern view

Code	**Description**
221	Parsing rules hive not found.
241	No such rule name in the site.xml.

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb eventpattern modify

Use the **vcb eventpattern modify** command to modify the pointer to the parsing rules.

Syntax

```
vcb eventpattern modify -patternid patternId -group parsing rules hive [-repository
parsing rules repository hive] [-rulename ruleName] -user username -password password
```

Description

The **vcb eventpattern modify** command modifies the pointer from the parsing-rules hive to the repository where the actual parsing rules are defined.

Usage Examples

```
vcb eventpattern modify
  -patternid 202
  -group cisco-syslog-product-parsing-rules
  -repository cisco-router-syslog-repository -user root -password admin
```

This example assumes that we are starting with the eventpattern with ID 202 that points to the cisco-syslog-repository (as shown in Usage Examples for the **vcb eventpattern add** command). In this example, we modify the repository for the eventpattern with ID 202 to the cisco-router-syslog-repository.

Note `Vcb eventpattern modify` cannot be performed on the **group** as this is the identifier for the pattern. You can modify only the **repository** or **rule name.**

Options

Table B-54 Options and Arguments—vcb eventpattern modify

Option ***Argument***	**Description**
patternid *patternId*	Unique integer to identify the supported event to a VNE.
group *parsing rules hive*	The hive in which this pattern is to be modified. The parsing-rules hives are generally scheme-specific. VNE-specific definitions can also be made using this hive.
repository *parsing rules repository hive*	(Optional) The hive where the actual parsing rules are defined (the trap/syslog repository). Enter the same hive that was specified when creating parsing rules registrations using the **vcb eventparsingrules add** command.
rulename *ruleName*	String that is used as a key name for event rule definition. **Note** Enter exactly the same string as the one that was specified when creating parsing rules registrations using the **vcb eventparsingrules add** command.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-55 Error Codes—vcb eventpattern modify

Code	**Description**
221	Parsing rules hive not found.
222	Parsing rules repository not found.
271	Pattern with ID not found

Note For the list of general VCB error codes, see General Error Codes, page B-2.

vcb eventpattern delete

Use the **vcb eventpattern delete** command to delete the parsing rule from the list of supported event patterns. Doing so does not delete the parsing rules in the repository file.

Syntax

```
vcb eventpattern delete -group parsing rule hive -patternid pattern ID -user username
-password password
```

Description

The **vcb eventpattern delete** command removes the pointer to the parsing rule defined in the repository file.

Usage Examples

```
vcb eventpattern delete -group cisco-syslog-parsing-rules -patternid 202 -user root
-password admin
```

Deletes the parsing rules pattern with ID 202. All registry entries added as a part of the **vcb eventpattern add** command will be removed from site.xml.

Options

Table B-56 Options and Arguments—vcb eventpattern delete

Argument	Description
patternid *pattern ID*	Unique integer to identify the supported event to a VNE.
group *parsing rules hive*	The hive in which this pattern is to be modified. The parsing-rules hives are generally scheme-specific. VNE-specific definitions can also be made using this hive.

Note For the list of global options, see Global Command Options, page B-2.

Error Codes

Table B-57 Error Codes—vcb event pattern delete

Code	Description
221	Parsing rules hive not found.
271	Pattern with ID not found

Note For the list of general VCB error codes, see General Error Codes, page B-2.

VCB CLI Reference: vcb eventarg Command

vcb eventarg view

Use the **vcb eventarg view** command to display event parsing rule arguments and descriptions.

Syntax

```
vcb eventarg view -user username -password password
```

Description

The **vcb eventarg view** command option displays all the VCB event parsing rules template variable arguments along with descriptions.

Troubleshooting Event Customization Using the VCB CLI

Errors that you receive from the VCB CLI are self-explanatory. Most errors make very clear what you need to do to correct the problem that has occurred. For example, if an event name already exists, you must enter a different event name. If an alarm ID or pattern ID is already in use, you should omit the related option and argument from your command and allow the VCB to generate a unique ID for you.

Note To get more information, add the **-debug** option to any **vcb** command; for more information, see Global Command Options, page B-2.

Errors that occur in the server are not as obvious. If a newly supported event does not appear in Prime Network Events or Prime Network Vision, you need to perform some troubleshooting, as follows:

Step 1 Ensure that the device is configured to send events to Prime Network gateway. Use any tool to snoop and check whether the simulated network events that you are sending are actually arriving at the Prime Network gateway. If not, fix the issue whether it is connectivity, firewall and so on, then proceed to next step.

For detailed information, see VCB CLI Reference: Events Commands, page B-31.

Step 2 Enable debug for event processing in the VNE. Execute the following commands for the AVM that has the VNE that you will be testing.

```
runRegTool.sh -gs localhost set 127.0.0.1
avm<avmid>/services/logger/log4j.category.com.sheer.metrocentral.framework.eventapplicatio
n.eventcorrelation.SendAlarmMessageUtil DEBUG
runRegTool.sh -gs localhost set 127.0.0.1
avm<avmid>/services/logger/log4j.category.com.sheer.metrocentral.framework.eventmanager.Ev
entManager DEBUG
runRegTool.sh -gs localhost set 127.0.0.1
avm<avmid>/services/logger/log4j.category.com.sheer.metrocentral.framework.eventapplicatio
n.parsing.ParsingApplication DEBUG
```

Step 3 Allow the VNE to come up, then open the log file for the AVM. Check whether the newly added pattern is being loaded at VNE startup. Look for an entry in the log file that is similar to the following text:

```
DEBUG [06 21 2010 12:19:59.524 IST] - ParsingApplication.buildRulesMatrix - pattern
ATMLC-6-CLOCKING with index 5001 in the registry is now mapped into index 123 in the
parsing application.
```

The above DEBUG statement includes both the rulename (ATMLC-6-CLOCKING with) and the pattern id (5001) of the newly added event. If a similar statement is not printed for the newly added event, go to Step 4; otherwise, go to Step 5.

Step 4 Review the **vcb eventparsingrules add** command that you used, checking whether you enabled the event using the **-enable** option. If the command was issued without the **-enable** option, delete the event parsing rules using the **vcb eventparsingrules delete** command and add the event parsing rules again, ensuring that you use with the **-enable** option.

Step 5 Check statically whether the links between event pattern, event parsing rules, and event are OK. To perform this check, use the **vcb eventpattern view** command with the **-full** option (see Example 3, page B-49). The output should display details of all the three customizations. A typo in the rulename, event type name, or event subtype name can prevent the links from being established and result in a partial display. For example, if the rulename in the **vcb eventpattern** command does not match that used in the **vcb eventparsingrules** command, only event pattern details will be displayed; details for event parsing rules and the event will not be displayed.

If the output is OK (that is, it includes details for all three customizations), go to Step 7. Otherwise, go to Step 6.

Step 6 Review the commands that have been issued and re-add or modify the customizations as required. Then go back to Step 5.

Step 7 After the static verification that you perform in step 5 succeeds, check whether the parsing itself is failing. Put a tail on the AVM log file and resend the simulated event. When parsing fails, the event is classed as a generic event. Log output similar to the following will appear.

```
DEBUG [06 21 2010 16:11:09.623 IST] - EventManager.filterEventApplications - Event
has been dropped by application
[com.sheer.metrocentral.framework.eventapplication.filter.GenericSyslogTypeFilterAp
p]

########## com.sheer.metrocentral.framework.eventapplication.types.EventData
########

  # Id              :  = 137611826381_1277116869542

  # Unique source ID:  = null

  # Type            :  = generic syslog

  # SubType         :  = generic syslog

  # SourceOID       :  = {[ManagedElement(Key=10.77.212.205)][Syslog]}

  # Event Time      :  = 1277116869542

  # Info          :  = 7.212.205 %FAN-3-FAN_OK: Fan 3 had earlier reported a rotation
error. It is ok now

  # CorrelationKeys: =

  #      CK=(MC.DA-10.77.212.205)-25:52:0:0 [16]

  # Adjacent XID   :  = null

  # Source IP interface:  = null


##################################################################################
```

Step 8 Open the log file and go backwards from the end of the file until you come to the place where logs pertaining to the actual parsing process are available. Search for the string 'Testing pattern: handle *rulename*', where rulename is the string you used in the **vcb eventpattern add** command. Here you will find logs that report the results of testing each rule. Identify the rule that failed as shown in the following log.

```
DEBUG [06 21 2010 16:11:09.622 IST] - ParsingApplication.processEvent - Exception
during parsing correlation rules, at pattern-125, rule-2

Stack:[(uniqueid=>3),(syslog=>7.212.205 %FAN-3-FAN_OK: Fan 3 had earlier reported a
rotation error. It is ok now),(subtypekey=>OK),(.prulescache=>[]),(counter=>0)]

Event Data :

  ############## com.sheer.metrocentral.framework.eventapplication.parsing.types

.RawSyslogEventData #############

  # Id             :  = 4311876356_1277116869490

  # Unique source ID:  = null

  # Type           :  = raw event

  # SubType        :  = raw syslog

  # SourceOID      :  = null

  # Event Time     :  = 1277116869494

  # Info         :  = 7.212.205 %FAN-3-FAN_OK: Fan 3 had earlier reported a rotation
error. It is ok now

  # syslog = 7.212.205 %FAN-3-FAN_OK: Fan 3 had earlier reported a rotation error. It
is ok now

##############################################################################

#################################

      java.lang.reflect.InvocationTargetException

        at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method) at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl
.java:39)

        at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAcce
ssorImpl.java:27)

        at java.lang.reflect.Constructor.newInstance(Constructor.java:513)

        at
com.sheer.metrocentral.framework.correlation.parsing.ChangeArgumentValue.execute
(ChangeArgumentValue.java:68)
```

In the above example, the parsing rule that failed is ChangeArgumentValue. The failure implies that the replacing rules that map the network event parameters to the Prime Network event subtypes are failing. Review the replacing_rules arguments used in the **vcb eventparsingrules** command and make the necessary changes.

The list of parsing rules (classes) and the corresponding option in vcb are given in the following table. Review the parameter values of the failing option and make appropriate changes.

Repeat the above steps until all errors are resolved and the event is parsed correctly and the Prime Network event is generated as expected.

VCB Template Reference

Use the information in this section to determine which template best matches the device, module, or event that you want to manage with Prime Network. Information in this section is also useful when you test the customizations that you have made.

Topics include:

- U-VNE Templates, page B-56
- Module Templates, page B-65
- Event Templates, page B-80

U-VNE Templates

Features, advantages, and limitations of template-based U-VNEs are template-dependent. The GenericUVNE template uses the same set of MIB-II based instrumentation for logical inventory discovery as is used by the Prime Network Generic SNMP VNE. The advantage of the U-VNE—created using the VCB and the GenericUVNE template—over the Generic SNMP VNE is that you can identify the device type for the U-VNE and further extend the U-VNE for additional event recognition using the VCB.

For more information, see the following sections:

- Methods for Creating U-VNEs—Overview, page 3-7
- Comparison of Generic SNMP VNEs, U-VNEs, and Developed VNEs, page 3-3
- GenericUVNE Template, page B-56

GenericUVNE Template

The GenericUVNE template is applicable to Cisco and non-Cisco NEs and supports event customization with event association to Managed Element only.

Use the GenericUVNE template to model any NE that is not currently supported by Prime Network. A U-VNE created this way is very similar to the Generic SNMP VNE. It provides basic information, such as the physical interfaces available on the device and their status, rudimentary logical modeling, and parsing of basic traps; see GenericUVNE—Supported Traps, page B-59. Using the VCB, however, you can configure additional traps and syslog recognition for a U-VNE created using the GenericUVNE template.

This U-VNE models NEs using SNMP MIB-II, which is the most generic and widely used management interface. This U-VNE does not consider the device vendor, device type, or software version of the NE that it models. Using the VCB, however, you can update device type attributes for a U-VNE created using the GenericUVNE template.

Table B-58 summarize the features and advantages of the GenericUVNE template.

Table B-58 GenericUVNE Summary

Features	Advantages	Limitations
• Same physical and logical inventory as a Generic SNMP VNE. • Applicable to Cisco and non-Cisco NEs. • User-defined device type attributes: – Device category—Determines the icon that is displayed. – Element type. • Event recognition, enabling Prime Network to forward events from unsupported devices to OSS applications.	When compared with a Generic SNMP VNE, this U-VNE provides: • Simplified trap and syslog recognition (using the VCB). • Application of soft properties to a specific U-VNE (as opposed to a device type).	Event association to Managed Element only

Note Expedite Legend—The Expedited column in the service event tables in this chapter can contain these values:
Y—Indicates that the service event is expedited by a syslog or trap generated by the device. This means that the syslog or trap causes the VNE to poll the device without waiting for the usual polling cycle, thus enabling quicker detection of the event.
N —Indicates that the service event is not expedited. The service event is not expedited.This means that the VNE will poll this device during the next regularly scheduled polling cycle.

Note In some of the following tables, attributes, protocols, technologies, etc. are listed as supported. Supported denotes that SNMP queries are made to the NE for those attributes, etc. Whether values are available in response to the queries depends on whether the instrumentation supported in the NE works.

See the following sections:

- GenericUVNE—Physical Inventory Model, page B-58
- GenericUVNE—Logical Inventory Model, page B-58
- GenericUVNE—Supported Traps, page B-59
- GenericUVNE—Supported Events, page B-59
- GenericUVNE—Limitations, page B-60
- GenericUVNE—Supported Topologies, page B-60
- GenericUVNEs—Supported Technologies, page B-60
- GenericUVNE—Supported Service Events, page B-65

GenericUVNE—Physical Inventory Model

A U-VNE created using the GenericUVNE template uses a static model for the device chassis. The rest of the physical inventory is modeled using the ifTable. Since modules are not modeled, this U-VNE creates a single generic module on which all of the physical interfaces reside.

Table B-59 describes which MIB tables are used to model the physical inventory components that are supported by a U-VNE created using the GenericUVNE template.

Table B-59 MIBs Used for Physical Inventory Model of GenericUVNE

Logical Component	MIB Table	Columns/Tables Used For Modeling
Interfaces	ifTable	• ifDescr • ifType • ifOperStatus
Ports		• ifOperStatus and ifAdminStatus
Port status	ifTable	• ifSpeed
Port speed	ifTable	• ifPhysAddress (Ethernet ports)
MAC address	ifTable	

Note Certain general properties on the managed element, such as system description, are modeled using the RFC1213-MIB.

GenericUVNE—Logical Inventory Model

Table B-60 describes which MIB tables are used to model the logical inventory components that are supported by a U-VNE created using the GenericUVNE template. Attributes in Table B-60 are taken from MIB-II.

Table B-60 MIBs Used for Logical Inventory Model of GenericUVNE

Logical Component	MIB Table	Columns/Tables Used For Modeling
IP Interfaces	ipAddrTable	• ipAdEntIfIndex • ipAdEntNetMask
ARP table	ipNetToMediaTable	• ipNetToMediaPhysAddress • ipNetToMediaType
Routing table	ipRouteTable	• ipRouteDest • ipRouteIfIndex • ipRouteNextHop • ipRouteType • ipRouteMask
Bridging table	dot1dTpFdbTable	—
Default bridge	dot1dBridge	• dot1dBaseBridgeAddress • dot1dBaseType

GenericUVNE—Supported Traps

A U-VNE created using the GenericUVNE template can parse the standard MIB-II and Bridge-MIB traps listed in Table B-61.

Table B-61 Supported Traps for GenericUVNE

Standard MIB-II Traps	
authenticationFailure	mplsTunnelReoptimized
bgpBackwardTransition	mplsTunnelRerouted
bgpEstablished	mplsTunnelUp
coldStart	ospfIfAuthFailure
entConfigChange	ospfIfConfigError
linkDown	ospfIfRxBadPacket
linkUp	ospfIfStateChange (down)
mplsL3VpnVrfDown	ospfIfStateChange (up)
mplsL3VpnVrfNumVrfRouteMaxThreshExceeded	ospfMaxAgeLsa
mplsL3VpnVrfRouteMidThreshExceeded	ospfNbrStateChange (down)
mplsL3VpnVrfUp	ospfNbrStateChange (up)
mplsLdpInitSessionThresholdExceeded	ospf-if-packet-retransmit
mplsLdpSessionDown	ospfOriginateLsa
mplsLdpSessionUp	ospfTxRetransmit
mplsTunnelDown	warmStart
Bridge-MIB Traps	
dot1dBaseBridgeAddress	dot1dBaseType

A U-VNE created using the GenericUVNE template can identify traps, but it cannot correlate them. This is because this U-VNE does not include the model entities required by higher trap parsing levels.

For example, if Prime Network receives an mplsTunnelDown trap from a device modeled with the GenericUVNE template, Prime Network can identify the Tunnel Down trap, but it cannot perform correlation on the trap. The reason is that th is U-VNE does not investigate tunnels, which means that there is no Device Component in the model to which Prime Network can attach a correlation flow.

For this U-VNE, event association is always to the **Managed Element**.

GenericUVNE—Supported Events

A U-VNE created using the GenericUVNE template supports the service events listed in Table B-62.

Table B-62 Supported Service Events for GenericUVNE

Event Name	**Supported**	**Expedite d**
Device Unreachable	Y	N
Discard Packets	Y	N

Table B-62 Supported Service Events for GenericUVNE (continued)

Event Name	Supported	Expedite d
Dropped Packets	Y	N
Port Flapping	Y	N
Port Down	Y	N

GenericUVNE—Limitations

A U-VNE created using the GenericUVNE template uses MIB2 to cover the widest possible range of NEs. Although MIB2 is a widely accepted industry standard, most network equipment vendors augment MIB2 with other Management Interfaces such as private MIBs, Telnet, XML, and so on. In addition, different vendors sometimes have different implementations of standard MIBs. As a result, even the limited model created by this U-VNE is dependent on the vendor’s adherence to general network management standards.

GenericUVNE—Supported Topologies

A U-VNE created using the GenericUVNE template supports the topologies listed in Table B-63.

Table B-63 Supported Topologies for GenericUVNE

Topology Type	Link Type	Supporte d
Ethernet	Ethernet	Y
Physical Layer	Ethernet	Y

GenericUVNEs—Supported Technologies

The following sections list the objects and attributes that are recognized on a U-VNE created using the GenericUVNE template, per technology:

- IP, page B-61
- Ethernet (IEEE 802.3), page B-62
- Base Logical Components, page B-63
- Common, page B-64

IP

Table B-64 lists the IP attribute support on a U-VNE created using the GenericUVNE template.

Note Table B-64 includes the supported technologies only.

Table B-64 IP Attribute Support on GenericUVNEs

Attribute	Supported
IMO Name—IIPInterface	
IP Address	Y
Subnetwork Mask	Y
IP Interface Addresses Array	
Interface Name	
Interface Description	Y
IP Interface State	Y
OSPF Interface Cost	
Broadcast Address	
MTU	
Lookup Method	
Address Resolution Type	
ARP Timeout	
Secured ARP	
ICMP Mask Reply	
IGMP Proxy	
HSRP Groups	
IP Multiplexing Table	
IANA Type	
Containing CTPs	
Contained CTPs	
IMO Name—IRoutingEntity	
Routing Table	Y
ARP Entity	Y
Routing Table Changes	
Name	Y
Logical Sons	Y

Table B-64 IP Attribute Support on GenericUVNEs (continued)

Attribute	Supported
IMO Name—IRoutingEntry	
Destination IP Subnet	Y
Next Hop IP Address	Y
Type	Y
Routing Protocol Type	Y
Outgoing Interface Name	Y
IMO Name—IARPEntity	
ARP Table	Y
IMO Name—IARPEntry	
IP Address	Y
MAC Address	Y
Port	Y
Entry Type	Y

Ethernet (IEEE 802.3)

Table B-65 lists the Ethernet (IEEE 802.3) attribute support on a U-VNE created using the GenericUVNE template.

Table B-65 Ethernet (IEEE 802.3) Attribute Support on GenericUVNEs

Attribute	Supported
IMO Name—IEthernet	
MAC Address	Y
Duplex Mode	
Output Flow Control	
Input Flow Control	
IANA Type	
Containing CTPs	
Contained CTPs	
Port Type	

Base Logical Components

Table B-66 lists the base logical attribute support on a U-VNE created using the GenericUVNE template.

Note Table B-66 includes the supported technologies only.

Table B-66 Base Logical Components Attribute Support on GenericUVNEs

Attribute	Supported
IMO Name—IManagedElement	
IP Address	Y
Communication State	Y
Investigation State	Y
Element Category	Y
Element Type and Key	Y
Device Name	Y
System Name	Y
System Description	Y
Up Time	Y
Software Version	Y
Vendor Identity	
Memory and CPU Usage	
DRAM Free	
DRAM Used	
Flash Device Size	
NVRAM Size	
Processor DRAM	
Sys Contact	
Sys Location	
Serial Number	
File Systems	
IMO Name—ISystemService	
Type	
Status	
Up Time	

Common

Table B-67 lists the common attribute support on a U-VNE created using the GenericUVNE template.

Note Table B-67 includes the supported technologies only.

Table B-67 Common Attribute Support on GenericUVNEs

Attribute	Supported
IMO Name—IPhysicalLayer	
Media Type	
Clocking Source	
Maximum Speed	Y
Is Internal Port	
Discarded Bandwidth	
Dropped Bandwidth	
Input Bandwidth	
Output Bandwidth	
Discarded and Received Input Data Counters	Y
Dropped and Forward Output Data Counters	Y
Administrative Status	Y
Operational Status	Y
Last Changed	Y
IANA Type	
Containing CTPs	
Contained CTPs	
Port Alias	
Location	
Sending Alarms	
Connector Description	
Part ID	
Connector Serial Num	
Product	
Status	
Managed	

Table B-67 Common Attribute Support on GenericUVNEs

Attribute	Supported
IMO Name—IBridgeEntry	
Destination MAC	Y
Outgoing Interface	Y

GenericUVNE—Supported Service Events

Table B-68 lists the supported service events on a U-VNE created using the GenericUVNE template.

Table B-68 Supported Service Events for GenericUVNE

Event Name	Supported	Expedited
Device Unreachable	Y	N
Discard Packets	Y	N
Dropped Packets	Y	N
Port Flapping	Y	N
Port Down	Y	N

Module Templates

Module templates define a set of port layers—from the connector at Layer 0 to encapsulation at Layer 2—that are applicable to a module. These templates ensure that each port is modeled with the correct port layer information based on the ifType obtained from the SNMP MIB output.

Module templates are applicable to standard modules only (not pluggable modules). You do not need to use a module template to add a pluggable module.

When adding support for a new module using the VCB, you must identify the module template that matches the capabilities of the module.

For example, the atm-default module template contains the following port layer definitiions, which are typical port layers for an OC3 ATM card:

- Layer 0—Fiber optic
- Layer 1—OC3
- Layer 2—ATM

These definitions make this template suitable for modules with ports that:

- Use fiber optic cable.
- Support the OC-3 data transfer rates over SONET.
- Use ATM encapsulation for transporting IP traffic between two peers.

Note You cannot add modules to a U-VNE created using the GenericUVNE template.

Module templates are collected into groups, such as the ciscophysicalspec2 group for Cisco modules. The information contained in the module specification files is summarized in the Module Groups and Module Specification Files, page B-66.

Note Module definitions that you create with the VCB are added to the module group that contains the template on which the definition is based.

After you obtain the module identifier and research the capabilities of the module, use Module Templates by Technology, page B-69 to identify the module template that best matches the module. You can then add support for the module using the VCB.

This section contains the following topics:

- Module Groups and Module Specification Files, page B-66
- Module Templates by Technology, page B-69

Module Groups and Module Specification Files

Note Unlike the modeling that Prime Network does for standard modules, Prime Network models only the ports for pluggable modules. The only module group for pluggable modules is the pluggable-ports-spec file. The remainder of this section applies to standard modules only (not pluggable modules).

A module group is the name of a vendor-specific module specification file that is stored in the Prime Network registry. A module specification file is an XML file that lists supported modules and other properties, such as port layers and sysOID. When you use **vcb module** commands to add, modify, or delete a module:

- You provide the name of a module specification file as an argument to the **-group** option. (For more information, see VCB CLI Command Reference: Standard Modules, page B-18.)
- The VCB modifies the module specification file: adding, updating, or deleting the module definition.

  Note The VCB allows you to update and delete only those modules that you added using the VCB.

Prime Network enables you to extend the following module specification files:

- ciscophysicalspec2
- ciscocatalyst3400spec
- cisco-catalyst-spec

Table B-69 summarizes the technologies that are supported and the module templates that are provided in the module specification files. For more information about a module template, use the link in the Technologies column.

Table B-69 Module Group Summary for Standard Modules

Module Group	Technologies	Template Names
ciscophysicalspec2	Ethernet (Fixed), page B-69	• A10GigaEthernet • ethernet-default-over-optic • ethernetDefault • gigaEthernet • EthernetChannelSwitchDefault
	Ethernet (Multiloader), page B-70	• 10Gigaethernet- • Gigaethernet-Fiber • GE-fiberoptic-ethernet-default • ethernetDefault-RJ45-or-Fiber2 • E1orGigabitTechnology2 • GE-over-OC12-pos-default • GE-over-OC3-pos-default3 • ethernet-or-oc-pos-default3 • ethernet-or-OC12- • pos-default3 • ethernet-or-oc48- • pos-default • DWDMA10GigaEthernet
	POS (Fixed), page B-72	• PPPdefaultOC48 • PPPdefault
	POS (Multiloader), page B-72	• POS-OC3-default • PPPdefaultOC12 • PPPdefaultOC192 • PPPdefaultOC3 • PPPdefaultOC768 • DWDMOC768

Table B-69 Module Group Summary for Standard Modules (continued)

Module Group	Technologies	Template Names
ciscophysicalspec2 (continued)	Channelized T1/E1 (Fixed), page B-73	• RJ45-T1E1Channelized • T1E1Channelized • T1E1Channelized-ATMorCEM • E3Default • E3Loader • E1Channelized • E1Default
	Channelized OCXX (Fixed), page B-74	• ChannelizedOC3 • ChannelizedOC12 • ChannelizedOC12xx • ChannelizedOCxx
	ATM (Fixed), page B-74	• atmDefault • atmOverOC12 • atm-over-e3ds3 • T1E1_ATM-IMA • ds1Default • ds3Default • adslDefault
	ATM (Multiloader), page B-75	• T3Channelized • T3Loader • layer2-over-ds1 • layer2-over-ds3 • layer2-over-ds3-over-bnc • pppOverDS3Default • layer2-over-e1 • HSSIDefault
	Multitechnology, page B-76	• 36xxMultiTechnologiesModuleDefault • 8xxMultiTechnologiesModuleDefault • MultiTechnologiesModuleLayers • MultiTechnologiesModuleDefault2
	Serial, page B-77	• PPPwithRJ11 • multichannelDefault • serialPPPDefault
	ISDN, page B-77	• ds1T1Default • BRIDefault

Table B-69 Module Group Summary for Standard Modules (continued)

Module Group	Technologies	Template Names
ciscophysicalspec2 (continued)	Generic, page B-78	• TSLineDefault • generic-port1 • voiceEMDefault
ciscocatalyst3400spec	Ethernet (Cisco Catalyst 3400), page B-79	• cisco-3400-MultiTechnologiesModuleDefault • cisco-3400-ethernetDefault- • RJ45-or-Fiber
cisco-catalyst-spec	Ethernet (Cisco Catalyst), page B-79	• EthernetDefault • FastEthernetDefault • GigaEthernetDefault • GigaEthernetOnCopper • giga-ethernet

Module Templates by Technology

This section presents module templates organized by technology:

- Ethernet (Fixed), page B-69
- Ethernet (Multiloader), page B-70
- POS (Fixed), page B-72
- POS (Multiloader), page B-72
- Channelized T1/E1 (Fixed), page B-73
- Channelized OCXX (Fixed), page B-74
- ATM (Fixed), page B-74
- ATM (Multiloader), page B-75
- Multitechnology, page B-76
- Serial, page B-77
- ISDN, page B-77
- Generic, page B-78
- Ethernet (Cisco Catalyst 3400), page B-79
- Ethernet (Cisco Catalyst), page B-79

Ethernet (Fixed)

Table B-70 lists module templates that support EthernetCSMA/CD at Layer 1 and a single connector type at Layer 0. When there is more than one Layer 2 option, Layer 2 is modeled based on transmission rate.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-70 Module Templates—Ethernet (Fixed)

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
A10GigaEthernet	Fiber optic	EthernetCSMA/CD	10 Gigabit Ethernet	• WS-SUP32-10GE-3B • 7600-ES+2TG
ethernet-default-over-optic	Fiber optic	EthernetCSMA/CD	• Ethernet • Fast Ethernet • Gigabit Ethernet	• WS-6700-DFC3B • WS-6700-DFC3BXL
ethernetDefault	RJ45	EthernetCSMA/CD	• Ethernet • Fast Ethernet • Gigabit Ethernet	• 8FE-TX-RJ45 • SPA-8X1FE-TX-V2
gigaEthernet	Fiber optic	EthernetCSMA/CD	Gigabit Ethernet	• WS-X4624-SFP-E • 7600-ES+3C
EthernetChannelSwitchDefault	RJ45	EthernetCSMA/CD	EtherChannel	NM-16ESW

Ethernet (Multiloader)

Table B-71 lists Ethernet module templates that support multiple options at more than one port layer.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Note In addition to Ethernet, some module templates in Table B-71 also support POS or DWDM ports. (See the footnotes for Table B-71.)

Table B-71 Module Templates—Ethernet (Multiloader)

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
10Gigaethernet-Gigaethernet-Fiber[1]	• Fiber optic • RJ45	EthernetCSMA/CD	• 10 Gigabit Ethernet • Gigabit Ethernet	• 76-ES+XC-20G3C • WS-X45-SUP6-E • WS-X4606-X2-E
GE-fiberoptic-ethernet-default[2]	RJ45	EthernetCSMA/CD	• Ethernet • Fast Ethernet	catalyst375024ME (cevModuleCat375024M)
	Fiber optic		• Gigabit Ethernet	
ethernetDefault-RJ45-or-Fiber[2]	• Fiber optic • RJ45	EthernetCSMA/CD	• Ethernet • Fast Ethernet • Gigabit Ethernet	• WS-X4232-RJ-XX • WS-X4524-GB-RJ45V

Table B-71 Module Templates—Ethernet (Multiloader) (continued)

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
E1orGigabitTechnology[2]	• RJ45 • Fiber optic	EthernetCSMA/CD	• Ethernet • Fast Ethernet • Gigabit Ethernet	Motherboard for 2941 (cevCpu2941)
	• RJ48	DS1/E1		
GE-over-OC12-pos -default[3]	Fiber optic	OC12	• PPP • HDLC	OSM-4OC12-POS-SI+
		EthernetCSMA/CD	• Gigabit Ethernet	
GE-over-OC3-pos-default[3]	Fiber optic	OC3	• PPP • HDLC	OSM-4OC3-POS-SI+
	RJ45	EthernetCSMA/CD	• Gigabit Ethernet	
ethernet-or-oc-pos-default[3]	Fiber optic	OC3	• PPP • HDLC	OSM-2+4GE-WAN+
		EthernetCSMA/CD	• Fast Ethernet • Gigabit Ethernet • 10 Gigabit Ethernet	
ethernet-or-OC12-pos-default[3]	Fiber optic	OC12	• PPP • HDLC	• OSM-2OC12-POS-SI • OSM-2OC12-POS-SI+
		EthernetCSMA/CD	• Fast Ethernet • Gigabit Ethernet • 10 Gigabit Ethernet	
ethernet-or-oc48-pos-default	Fiber optic	OC48	• PPP • HDLC	OSM-1OC48-POS-SI+
		EthernetCSMA/CD	• Fast Ethernet • Gigabit Ethernet • 10 Gigabit Ethernet	
DWDMA10GigaEthernet[4]	Fiber optic	• DWDM • EthernetCSMA/CD	10 Gigabit Ethernet	• 76-ES+XT-2TG3CXL • 76-ES+XT-4TG3C

[1] The connector type is modeled as RJ45 only for a Gigabit Ethernet port that is not pluggable. The connector type for other Gigabit Ethernet and 10 Gigabit Ethernet ports is modeled as fiber optic.

[2] The connector type is modeled based on transmission rate.

[3] This module template supports either Ethernet or POS ports.

[4] This module template supports either Ethernet or DWDM ports.

POS (Fixed)

Table B-72 lists POS module templates that support fixed Layer 0 (fiber optic) and Layer 1 (OC3, OC12, or OC48) options.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-72 Module Templates—POS (Fixed)

Template Name	Maximum Transmission Rate (Mbps) Supported	Layer 0	Layer 1	Layer 2	Example Modules
PPPdefaultOC48	2488.32	Fiber optic	OC48	• PPP • HDLC • Frame relay	• SFP-OC48-IR1 • gsr-e-qoc48-sm-lr-sc • 16OC48-POS/DPT
PPPdefault	155.52	Fiber optic	OC3	• PPP • HDLC • Frame relay	• GSR-SFC6 • GSR-CSC

POS (Multiloader)

Table B-73 lists templates for POS modules with multiple Layer 2 options.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-73 Module Templates—POS (Multiloader)

Template Name	Maximum Transmission Rate (Mbps) Supported	Layer 0	Layer 1	Layer 2	Example Modules
POS-OC3-default	155.52	Fiber optic	OC3	• PPP • HDLC • Frame Relay	—
PPPdefaultOC12	622.08	Fiber optic	OC12	• PPP • HDLC • Frame Relay	SPA-8XOC12-POS
PPPdefaultOC192	9,953.28	Fiber optic	OC192	• PPP • HDLC	SPA-OC192POS-LR

Table B-73 Module Templates—POS (Multiloader) (continued)

Template Name	Maximum Transmission Rate (Mbps) Supported	Layer 0	Layer 1	Layer 2	Example Modules
PPPdefaultOC3	155.52	Fiber optic	OC3	• PPP • HDLC • Frame Relay	• SFP-OC3-SR • SFP-OC3-IR1
PPPdefaultOC768	39,813.12	Fiber optic	OC768	• PPP • HDLC • Frame Relay	1OC768-ITU/C
DWDMOC768	39,813.12	Fiber optic	OC768	• PPP • HDLC	—

Channelized T1/E1 (Fixed)

Table B-74 lists templates for channelized T1/E1 modules where port layers are fixed.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-74 Module Templates—Channelized T1/E1 (Fixed)

Template Name	Maximum Transmission Rate (Mbps) Supported	Layer 0	Layer 1	Layer 2	Example Modules
RJ45-T1E1Channelized	1.544	RJ45	T1E1	—	SPA-8XCHT1/E1
T1E1Channelized	1.544	RJ48	T1E1	—	• NM-2CE1T1-PRI • PA-MC-4T1
T1E1Channelized-ATMorCEM[1]	1.544	RJ48	T1E1	—	HWIC-4T1/E1
E3Default	44.736	BNC	DS3	—	ESR-8E3/DS3
E3Loader	44.736	BNC	DS3	—	—
E1Channelized	1.544	RJ48	E1	—	PA-8CE1
E1Default	1.544	RJ45	E1T1	—	VWIC2-1MFT-T1E1

[1] Layer 2—ATM or CEM—is built at runtime.

Channelized OCXX (Fixed)

Table B-75 lists templates for Optical Carrier modules where port layers are fixed.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-75 Module Templates—Channelized OCXX (Fixed)

Template Name	Maximum Transmission Rate (Mbps) Supported	Layer 0	Layer 1	Layer 2	Example Modules
ChannelizedOC3	155.52	Fiber optic	OC3	—	—
ChannelizedOC12	622.08	Fiber optic	OC12	—	—
ChannelizedOC12xx	622.08	Fiber optic	OC12	—	SPA-1XCHOC12/DS0
ChannelizedOCxx	155.52	Fiber optic	OC3	—	• SPA-1XCHSTM1/OC3 • SPA-CHOC3-CE-ATM

ATM (Fixed)

Table B-76 lists templates for ATM modules where port layers are fixed.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-76 Module Templates—ATM (Fixed)

Template Name	Maximum Transmission Rate (Mbps) Supported	Layer 0	Layer 1	Layer 2	Example Modules
atmDefault	155.52	Fiber optic	OC3	ATM	GSR-SFC12410
atmOverOC12	—	Fiber optic	OC12	ATM	• SPA-1XOC12-ATM-V2 • SPA-1XOC12-ATM
atm-over-e3ds3	—	RJ48	DS3	ATM	—
T1E1_ATM-IMA	1.544	RJ48	T1E1	ATM	—
ds1Default	—	BNC	DS1	ATM	—
ds3Default	—	BNC	DS3	ATM	PA-A3-T3
adslDefault	—	RJ11	ADSL	ATM	• WIC-1SHDSL • WIC-1ADSL • WIC-1ADSL-DG

ATM (Multiloader)

Table B-77 lists templates for ATM modules with multiple Layer 2 options.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-77 Module Templates—ATM (Multiloader)

Template Name	Maximum Transmission Rate (Mbps) Supported	Layer 0	Layer 1	Layer 2	Example Modules
T3Channelized[1]	—	BNC	DS3	• PPP • HDLC • Frame relay • ATM	• SPA-2XCT3/DS0 • PA-2T3/E3-EC
T3Loader	—	BNC	DS3	• PPP • HDLC • Frame relay	—
layer2-over-ds1	1.544	RJ48	DS1	• PPP • HDLC • Frame relay • ATM	VWIC-2MFT-T1-DIR
layer2-over-ds3	44.736	RJ48	DS3	• PPP • HDLC • Frame relay • ATM	• NM-1A-E3 • NM-1T3/E3
layer2-over-ds3-over-bnc	44.736	BNC	DS3	• PPP • HDLC • Frame relay • ATM	—
pppOverDS3Default	44.736	BNC	DS3	• PPP • HDLC	• copper-6ds3 • copper-12ds3 • 2DS3-SMB • NM-4T

Table B-77 Module Templates—ATM (Multiloader) (continued)

Template Name	Maximum Transmission Rate (Mbps) Supported	Layer 0	Layer 1	Layer 2	Example Modules
layer2-over-e1	1.544	RJ48	E1	• PPP • HDLC • Frame relay • ATM	NM-1CE1T1-PRI
HSSIDefault	—	DB50	HSSI	• PPP • HDLC • Frame relay	—

[1] This module template supports both full and channelized T3 and ATM over T3.

Multitechnology

Table B-78 lists multitechnology templates, including the following, which support modules where the connector type is not determined:

- MultiTechnologiesModuleLayers
- MultiTechnologiesModuleDefault

Note Do not use these templates unless no other template matches the modules to be added.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-78 Module Templates—Multitechnology

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
36xxMultiTechnologiesModuleDefault[1]	RJ45	• EthernetCSMA/CD	• Fast Ethernet • Gigabit Ethernet	—
		• Serial	• PPP • HDLC • Frame relay	
8xxMultiTechnologiesModuleDefault	RJ45	• EthernetCSMA/CD	• Fast Ethernet • Gigabit Ethernet	—
		• Serial	• PPP • HDLC • Frame relay	

Table B-78 Module Templates—Multitechnology (continued)

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
MultiTechnologiesModuleLayers[2]	• RJ11 • RJ45 • RJ48 • Fiber optic • DB60	• EthernetCSMA/CD • DS1 • E1 • OC3 • ADSL • Serial	• Fast Ethernet • Gigabit Ethernet • PPP • HDLC • Frame relay • ATM	OSM-1CHOC12/T3-SI
MultiTechnologiesModuleDefault[2]	• RJ11 • RJ45 • RJ48 • Fiber optic • DB60	• EthernetCSMA/CD • DS1 • E1 • OC3 • ADSL • Serial	• Fast Ethernet • Gigabit Ethernet • PPP • HDLC • Frame relay • ATM	gsr-sfc16-oc192

[1] This module template primarily supports Cisco 3600 series modules.

[2] Use this template only when no other template matches the modules to be supported.

Serial

Table B-79 lists templates that provide support for modules with serial interfaces when the information for Layer 1 is not clear.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-79 Module Templates—Serial

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
PPPwithRJ11	RJ11	Serial	PPP	WIC-1AM-V2
multichannelDefault	RJ48	Serial	PPP	—
serialPPPDefault	• RJ45 • RJ48 • Fiber optic • DB60 • Generic connector	Serial	• PPP • HDLC • Frame relay	• HWIC-4T • NM-2W • WIC-SERIAL-1T

ISDN

Table B-80 lists templates for ISDN modules.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-80 Module Templates—ISDN

Template Name	Maximum Transmission Rate (kbps) Supported	Layer 0[1]	Layer 1	Layer 2	Example Modules
ds1T1Default	—	• RJ45 • RJ48 • DB60 • Fiber optic	• DS1 • E1 • EthernetCSMA/CD	ISDN	—
BRIDefault	64	RJ45	ISDN layer 1	ISDN layer 2	WIC-1B-U-V2

[1] The connector type is modeled based on the type of port.

Generic

Table B-81 lists generic templates. Use them to configure modules for technologies that Prime Network does not support.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table B-81 Module Templates—Generic

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
TSLineDefault[1]	Generic connector	Generic Layer 1	—	• NM-16A • NM-32A
generic-port[1]	Generic connector	Generic Layer 1	Generic Layer 2	—
voiceEMDefault[2]	RJ45	Generic Layer 1	Generic Layer 2	• VIC-EM • VIC-4VP-FXS-DID

[1] This template provides support for modules whose technologies are not currently supported in Prime Network.

[2] This template provides support for voice modules.

Ethernet (Cisco Catalyst 3400)

Table B-82 lists templates that support modules for the Cisco Catalyst 3400 device group.

Module Group

These templates are defined in the ciscocatalyst3400spec module group.

Table B-82 Module Templates—Ethernet (Cisco Catalyst 3400)

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
cisco-3400-MultiTechnologiesModuleDefault **Note** Not recommended for 10/100/1000 Ethernet ports. Instead, use cisco-3400-ethernetDefault-RJ45-or-Fiber.	RJ45	EthernetCSMA/CD	Ethernet	3400 fixed modules
cisco-3400-ethernetDefault-RJ45-or-Fiber	• RJ45 • Fiber optic	EthernetCSMA/CD	• Ethernet • FastEthernet • GigaEthernet	3400 fixed modules

Ethernet (Cisco Catalyst)

Note For modules in the Cisco Catalyst 3400 device group, see Ethernet (Cisco Catalyst 3400).

Table B-83 lists templates that support modules for Cisco Catalyst devices.

Module Group

These templates are defined in the cisco-catalyst-spec module group.

Table B-83 Module Templates—Ethernet (Cisco Catalyst)

Template Name	Layer 0	Layer 1	Layer 2	Example Modules
EthernetDefault	RJ45	EthernetCSMA/CD	Ethernet	• WS-C3560G-24TS • WS-SUP720-3BXL
FastEthernetDefault	RJ45	EthernetCSMA/CD	FastEthernet	• WS-X4148-RJ • ws-c2924-xl-v
GigaEthernetDefault	RJ45	EthernetCSMA/CD	GigaEthernet	• WS-F6K-MSFC2A • OSM-2+4GE-WAN+
GigaEthernetOnCopper	UTP	EthernetCSMS/CD	GigaEthernet	Catalyst 6500 Supervisor Module 720 base board
giga-ethernet	Fiber optic	EthernetCSMA/CD	GigaEthernet	• cat6k-wsx-6066-slb-apc • wsx6ksup1a2ge • wsx6ksup22ge

Event Templates

Event templates work together to extract information from a syslog or a trap and to generate the keys and the location ID for associating a Prime Network event with a managed element device component. For more information, see the following sections:

- Terminology Used in Event Templates, page B-80
- Supported Interface Types, page B-80
- Event Templates Functional Summary, page B-81
- Event Templates Input Summary—Required and Optional Input, page B-83

For information about specific template types, see the following sections:

- Event Identification Templates, page B-84
- Unique ID Templates—for Traps Only, page B-86
- Event Subtype Templates, page B-87
- Entity ID Templates—for Traps Only, page B-89
- Entity Key Templates , page B-90
- Prime Network Event Templates, page B-92

Terminology Used in Event Templates

The following terms are used in the tables in this section.

Entity ID

The entity ID identifies the entity in the VNE with which to associate the event. For example, for interface-based events, ifIndex or ifName can be used as the entity ID.

Unique ID

The unique ID is used to create a unique location for the event when association to the exact entity is not possible. For example, when associating BGP traps to the Managed Element, the neighbor IP address can be used as the unique ID.

Supported Interface Types

The VCB can automatically identify the following interface types and associates events to them:

- Ether Channel
- GRE Tunnel
- DSO Bundle
- MPLSTunnel
- IMA Group
- MLPPP
- CEM Group
- IpInterface (Loopback, Vlan, all other subinterfaces)

For other interface types, the VCB associates the event to the layer 1 device component.

Event Templates Functional Summary

Event templates extract information and generate keys to associate an event with the correct VNE or U-VNE component. Table B-84 lists event templates (by type) and explains what each template does.

Table B-84 Event Template Functions

Template	Performs This Function for a...	
	Syslog	**Trap**
Event Identification Templates—Mandatory. Use one.		
snmp-trap-identification	—	Extracts information that identifies an event
syslog-identification	Extracts: • Information that identifies an event • Event subtypes • Entity ID • Unique ID	—
Event Subtype Templates—Optional. Use one if there are subtypes for the event.		
snmp-trap-subtype-from-oid	—	• Extracts subtype information from the event • Maps the subtype values to the event subtypes defined in Prime Network
snmp-trap-subtype-from-trapoid		
snmp-trap-subtype-from-value		
syslog-subtype-from-expression	Maps the subtype values—extracted by the syslog-identification template—to the event subtypes defined in Prime Network	—
Unique ID Templates—Optional. Use one when you cannot associate the event to an exact entity.		
snmp-trap-identifier-from-oid	—	Extracts the unique ID
snmp-trap-identifier-from-value		
Entity ID Templates—Optional. Use one when you can associate the event to an exact entity.		
snmp-trap-entity-from-oid	—	Extracts the entity ID
snmp-trap-entity-from-value		

Table B-84 Event Template Functions (continued)

	Performs This Function for a...	
Template	**Syslog**	**Trap**
Entity Key Templates—Mandatory. Use one.		
create-managedelement-key	• Creates the device component key • Associates the event with the Managed Element device component	
create-interface-key-from-ifindex	—	Creates the interface device component key from the ifIndex, using the entity ID that was extracted by an entity ID template (snmp-trap-entity-from-oid or snmp-trap-entity-from-value) Associates the event with the appropriate interface layer; see Supported Interface Types, page B-80
create-interface-key-from-ifname	• Creates the interface device component key from the ifname, using the entity ID that was extracted by: – syslog-identification template for a syslog – an entity ID template (snmp-trap-entity-from-oid or snmp-trap-entity-from-value) for a trap • Associates the event with the appropriate interface layer; see Supported Interface Types, page B-80 **Note** This template is more frequently used with syslogs than with traps.	
Prime Network Event Templates—Mandatory. Use one.		
create-ana-trap-event	—	Creates both of the following: • A unique location for the event, based on the entity ID (device component key) or the unique ID (extracted by an entity ID or a unique ID template) • Prime Network event, using subtypes extracted by an event subtype template: – snmp-trap-subtype-from-oid – snmp-trap-subtype-from-trapoid – snmp-trap-subtype-from-value
create-ana-syslog-event	Creates both of the following: • A unique location for the event, based on entity ID (device component key) or unique ID (extracted by the syslog-identification template). • Prime Network event, using subtypes extracted by the syslog-identification template	—

For the input required for each template, see Event Templates Input Summary—Required and Optional Input, page B-83.

Event Templates Input Summary—Required and Optional Input

Table B-85 summarizes event templates and the mandatory and optional input arguments for them.

Table B-85 Event Template Variables

Templates	Variable Name	Variable Type	Variable Description
Event Identification Templates			
syslog-identification	expression	Mandatory	Regular expression to match against the incoming syslog message.
	testmessage	Optional	String that is an example of the actual syslog message.
snmp-trap-identification	oid	Mandatory	SNMP trap OID.
Unique ID Templates			
snmp-trap-identifier-from-oid	inOID	Mandatory	Partial string that uniquely identifies the required OID in the varbind list.
	index	Optional	Used to create a unique location ID for the event when association to the exact entity is not possible. Default value is 1.
snmp-trap-identifier-from-value	inOID	Mandatory	Partial string that uniquely identifies the required OID in the varbind list.
Event Subtype Templates			
snmp-trap-subtype-from-oid	inOID	Mandatory	Partial string that uniquely identifies the required OID in the varbind list.
	replacing-rules		Defines mapping between event subtype and value in the trap that indicates the subtype. The subtype names provided in replacing-rules must match the subtype names used to add the event using the **vcb event add** command. To view event subtypes, use the **vcb event view** command.
	index	Optional	Used to create a unique location ID for the event when association to the exact entity is not possible. Default value is 1.
snmp-trap-subtype-from-trapoid	replacing-rules	Mandatory	Defines the mapping between event subtype and value in the trap that indicates the subtype.
snmp-trap-subtype-from-value	inOID	Mandatory	Partial string that uniquely identifies the required OID in the varbind list.
	replacing-rules		Defines the mapping between event subtype and value in the trap that indicates the subtype.
syslog-subtype-from-expression	replacing-rules	Mandatory	Defines the mapping between event subtype and value in the trap that indicates the subtype.

Table B-85 Event Template Variables (continued)

Templates	Variable Name	Variable Type	Variable Description
Entity ID Templates			
snmp-trap-entity-from-oid	inOID	Mandatory	Partial string that uniquely identifies the required OID in the varbind list.
	index	Optional	Used to create a unique location ID for the event when association to the exact entity is not possible. Default value is 1.
snmp-trap-entity-from-value	inOID	Mandatory	Partial string that uniquely identifies the required OID in the varbind list.
Entity Key Templates			
create-interface-key-from-ifindex	—	—	—
create-interface-key-from-ifname	—	—	—
create-managedelement-key	—	—	—
Prime Network Event Templates			
create-ana-trap-event	type	Mandatory	Event name, a string that must match the event name used to create the Prime Network event using the **vcb event add** command.
	subtype	Mandatory for INFO events only	Event subtype name, a string that must match the subtype name used to create the Prime Network event using the **vcb event add** command.
create-ana-syslog-event	type	Mandatory	Event name, a string that must match the event name used to create the Prime Network event using the **vcb event add** command.
	subtype	Mandatory for INFO events only	Event subtype name, a string that must match the subtype name used to create the Prime Network event using the **vcb event add** command.

Event Identification Templates

Event identification templates are mandatory. You must use one of these templates:

- snmp-trap-identification, page B-84
- syslog-identification, page B-85

snmp-trap-identification

This template supports SNMP V1, V2, and V3 traps. Any incoming SNMP V1 traps are converted automatically to SNMP V2 and then parsed as SNMP V2 traps. (The first rule in this template is the conversion rule.)

Mandatory Input

oid—The trap OID. Table B-86 describes how to format the OID for different traps.

Table B-86 Template Input—snmp-trap-identification

If the OID...	Format Input Like This...	Example Input String
Is for a V1 trap	Supply the enterprise OID appended by 0 and then by the specific type.	—
Contains subtype information as these do: • mplsLdpLibLspUp - 1.3.6.1.4.1.9.10.65.2.0.5 • mplsLdpLibLspDown - 1.3.6.1.4.1.9.10.65.2.0.6	Remove subtype information from the input string.	1.3.6.1.4.1.9.10.65.2.0
Is an informational trap as this one is: mplsLdpPathVectorLimitMismatch - 1.3.6.1.4.1.9.10.65.2.0.2	Supply the entire OID.	1.3.6.1.4.1.9.10.65.2.0.2
Does not contain subtype information; however, subtype information is contained in one of the varbinds: demandNbrLayer2Change - 1.3.6.1.4.1.9.9.26.2.0.3 The subtype for this trap is a varbind: isdnLapdOperStatus.	Supply the entire OID.	1.3.6.1.4.1.9.9.26.2.0.3

syslog-identification

This template supports syslogs. This template contains a single rule that extracts not only event identification information, but also subtype, unique ID, and entity ID when they are available for the event.

Mandatory Input

expression—A regular expression to be matched against the incoming syslog message.

The input string need not be a complete regular expression. It can be a partial syslog message with the parameters in which you are interested marked using keywords. Replace event-specific parameters in the syslog message with the following keywords:

- %%subtypekey%%—Use when the substring represents an event subtype.
- %%uniqueid%%—Use when the substring represents a parameter that uniquely identifies the event.
- %%entityid%%—Use when the substring represents the associated entity of the event.

For example, for the following syslogs:

%C6KENV-4-CLOCKFAILED: clock [dec] failed

%C6KENV-4-CLOCKOK: clock [dec] operational

you could provide the following input string:

%C6KENV-4-CLOCK%%subtypekey%%: clock %%uniqueid%%

There are no %%entityid%% parameters in this example, because this syslog must be associated with the ManagedElement device component.

The VCB uses the input string to automatically create a regular expression: ".*%C6KENV-4-CLOCK(\S+):clock (\S+).*"

There are instance when we require more than one input to uniquely identify the entity in the VNE. VCB allows you to subscript the keywords %%entityid%% and %%uniqueid%% with integers in order to specify more than one input that constitutes the entityid, for example, %%entityid1%%, %%entityid2%%.

For example, for EFP syslogs:

%ETHER_SERVICE-6-UPDOWN: Service instance 111 on interface GigabitEthernet10/0/3 changed to down.

Here, service instance id and interface name are inputs required to uniquely identify the EFP instance in the VNE. Read the template documentation to determine which variable should be marked as entityid1 and which should be marked entityid2.

Optional Input

testmessage—An input string that supplies an example of the actual syslog message. For example:

%C6KENV-4-CLOCKFAILED: clock 1 failed

If supplied, VCB checks this test message against the automatically created regular expression.

Input Format

```
-syslog_identification_expression regular_expression
-syslog_identification_testmessage message
```

Unique ID Templates—for Traps Only

Note For syslogs, unique ID information is extracted by the event identification template. For more information, see Event Identification Templates, page B-84.

The unique ID differentiates a particular instance of an event from other events of the same type. This parameter is required when:

- It is not possible to associate the event to the appropriate device component in the VNE for some reason, possibly one of the following:
  - The device component is not modeled due to lack of technology support.
  - The corresponding key generation template is not available.
- A unique ID is not required at this time. For example, when Prime Network associates interface traps to a managed element device component, the ifIndex or ifName is the unique ID. Prime Network automatically appends this unique ID to the location, thereby creating a unique location for each interface despite associating the event with a common entity, the managed element device component.

These templates differ in the way the information is extracted from the trap; select and use only one of the following:

- snmp-trap-identifier-from-oid, page B-87
- snmp-trap-identifier-from-value, page B-87

snmp-trap-identifier-from-oid

The rules in this template extract the unique identifier from one of the OIDs in the varbind list of the trap.

Mandatory Input

inOID—A partial string that uniquely identifies the required OID in the varbind list.

Optional Input

index—The location of the required value from the end of the OID. Default value is 1.

Input Format

```
-snmp_trap_identifier_from_oid_inOID OID
-snmp_trap_identifier_from_oid_index indexValue
```

snmp-trap-identifier-from-value

The rules in this template extract the unique identifier from the value of one of the OIDs in the varbind list.

Mandatory Input

inOID—A partial string that uniquely identifies the required OID in the varbind list.

Input Format

```
-snmp_trap_identifier_from_value_inOID OID
```

Event Subtype Templates

Use an event subtype template when configuring a trap or a syslog that includes event subtypes. For example, you should use an event subtype template when events arrive as a multistatus set or in asserted and cleared pairs, as is the case with link status traps. Link status traps send Link Down and Link Up traps, two subevents that are related to the same event:

- An asserted event indicates that the link is down.
- A clearing event indicates that the link status has changed to up.

Event subtype templates do the following:

- Extract event subtype information for traps. (For syslogs, event subtype information is extracted by the syslog-identification template; see syslog-identification, page B-85.)
- Map the subtype value to the event subtype name defined in Prime Network.

When subtypes exist for a trap, use one of these event subtype templates:

- snmp-trap-subtype-from-oid, page B-88
- snmp-trap-subtype-from-trapoid, page B-88
- snmp-trap-subtype-from-value, page B-88

When subtypes exist for a syslog, use this template:

- syslog-subtype-from-expression, page B-89.

snmp-trap-subtype-from-oid

The rules in this template extract event subtype information from one of the OIDs in the varbind list of the trap.

Mandatory Input

inOID—A partial string that uniquely identifies the required OID in the varbind list.

replacing-rules—Replacing rules define the mapping between the event subtype and the value in the trap that indicates the subtype.

The format for a rule is *value-event subtype*. The hyphen between the value and the event subtype is mandatory. Rules must be comma-separated.

Note Supply the same event subtype that was defined for the event with the **vcb event add** command. Use the **vcb event view** command to obtain a list of subtypes.

Optional Input

index—The location of the required value from the end of the OID. Default value is 1.

Input Format

```
-snmp_trap_subtype_from_oid_inOID OID
-snmp_trap_subtype_from_oid_index indexValue
-snmp_trap_subtype_from_oid_replacing_rules value-subtype, value-subtype, value-subtype
```

Note The hyphen between value and subtype is required.

snmp-trap-subtype-from-trapoid

The rules in this template extract event subtype information from one of the OIDs in the varbind list of the trap.

Mandatory Input

replacing-rules—Replacing rules define the mapping between the event subtype and the value in the trap that indicates the subtype.

The format for a rule is *value-event subtype*. Rules must be comma-separated.

Note Supply the same event subtype that was defined for the event with the **vcb event add** command. Use the **vcb event view** command to obtain a list of subtypes.

Input Format

```
-snmp_trap_subtype_from_trapoid_replacing_rules value-subtype, value-subtype,
value-subtype
```

snmp-trap-subtype-from-value

The rules in this template extract the event subtype information from the value of one of the OIDs in the varbind list.

Mandatory Input

- **inOID**—A partial string that uniquely identifies the required OID in the varbind list.
- **replacing-rules**—Replacing rules define the mapping between the event subtype and the value in the trap that indicates the subtype.

The format for a rule is *value-event subtype*. Rules must be comma-separated.

Note Supply the same event subtype that was defined for the event with the **vcb event add** command. Use the **vcb event view** command to obtain a list of subtypes.

Input Format

-snmp_trap_subtype_from_value_inOID *inOID*

-snmp_trap_subtype_from_value_replacing_rules *value-subtype, value-subtype, value-subtype*

syslog-subtype-from-expression

The rules in this template extract the event subtype information from the value of one of the OIDs in the varbind list.

Mandatory Input

replacing-rules—Replacing rules define the mapping between the event subtype and the value in the trap that indicates the subtype.

The format for a rule is *value-event subtype*. Rules must be comma-separated.

Note Supply the same event subtype that was defined for the event with the **vcb event add** command. use the **vcb event view** command to obtain a list of subtypes.

Input Format

-syslog_subtype_from_expression_replacing_rules *value-subtype, value-subtype, value-subtype*

Entity ID Templates—for Traps Only

Note For syslogs, entity ID information is extracted by an event identification template. For more information, see Event Identification Templates, page B-84.

The entity ID specifies the device component to which the event should be associated in the VNE. These templates differ in the way that the information is extracted from the trap:

- snmp-trap-entity-from-oid, page B-89
- snmp-trap-entity-from-value, page B-90

snmp-trap-entity-from-oid

The rules in this template extract the entityID from the one of OIDs in the varbind list of the trap.

Mandatory Input

inOID—A partial string that uniquely identifies the required OID in the varbind list

Optional Input

index—The location of the required value from the end of the OID. Default value is 1.

Input Format

```
-snmp_trap_entity_from_oid_inOID inOID
```

snmp-trap-entity-from-value

The rules in this template extract the entity ID from the value of one of the OIDs in the varbind list.

Mandatory Input

inOID—A partial string that uniquely identifies the required OID in the varbind list

Input Format

```
-snmp_trap_entity_from_value_inOID inOID
```

Entity Key Templates

Selecting a template from this category is mandatory. Entity key templates use the entity ID information—extracted using other templates—to generate a key to uniquely identify the device component in the VNE. The event is later associated with the corresponding device component.

For templates that extract the entity ID, see syslog-identification, page B-85 and Entity ID Templates—for Traps Only, page B-89.

Select one of the following templates:

- create-interface-key-from-ifindex, page B-90
- create-interface-key-from-ifname, page B-90
- create-managedelement-key, page B-91
- create-efp-key-from-ifname-serviceid, page B-91
- create-logical-container-key , page B-91
- create-moduleDC-key-given-entPhysicalIndex, page B-91
- create-moduleDC-with-slotSubslot-value-key, page B-91
- create-pw-interface-key-from-tunnelindex, page B-92

create-interface-key-from-ifindex

This template creates the interface device component key from the ifIndex and associates the event with the appropriate interface layer.

No input is required.

create-interface-key-from-ifname

This template creates the interface device component key from an ifName and associates the event with the appropriate interface layer.

No input is required.

create-managedelement-key

This template creates the managed element device component key, associating the event with the managed element device component.

No input is required.

create-efp-key-from-ifname-serviceid

This template creates the EFP DC key from the Service Instance+ifName.

Note The service instance ID should be marked as %%entityid1%% and interface name as %%entityid2%% in the syslog expression given by the user.

The event will be associated with particular EFP DC. For example, consider the following:

Syslog Feb 8 11:04:18 MSK: %ETHER_SERVICE-6-UPDOWN: Service instance 214 on interface TenGigabitEthernet3/3 changed to up

The syslog expression for this should be given as %ETHER_SERVICE-6-UPDOWN: Service instance %%entityid1%% on interface %%entityid2%% change to %%subtypekey%%.

No input is required.

create-logical-container-key

This template associates syslog/trap events to the designated containers, as preferred.

Containers list: CfmService,BfdService,MPBgp,REPService,StpService,SbcService,EthernetLMI,ISISSystem,LSE,ClockService. Provide the desired logical container string, from the containers list.

Requires mandatory user input.

create-moduleDC-key-given-entPhysicalIndex

This template associates trap events to the designated moduleDC, when its observed entPhysicalIndex of Entity-MIB, through mib instrumentation queries from the device; Is given as the input for the desired module entity.

No input is required.

create-moduleDC-with-slotSubslot-value-key

This template associates syslog events to the corresponding module, knowing the residing slot number. For example, Syslog message "%OIR-6-REMCARD: Card removed from slot 4, interfaces disabled" USER_INPUT_MANDATORY <- %OIR-6-REMCARDCARD: Card removed from slot%%entityid%%, interfaces disabled entityid <- 4

No input is required.

create-pw-interface-key-from-tunnelindex

This template associates trap events to the designated pseudowire tunnel interface. To achieve this, provide the appropriate oid of the var bind for the trap to be associated with this pseudowire tunnel interface while providing inputs for "snmp-trap-entity-from-oid" template. The tunnel interface index provided by "snmp-trap-entity-from-oid" template, becomes the automatic input to the current "create-pw-interface-key-from-tunnelindex" template.

No input is required.

Prime Network Event Templates

These templates create a unique location ID for the event using the device component key and unique ID information created in the previous rules.

create-ana-trap-event

Template to create a unique location ID for a Prime Network trap event.

Mandatory Input

type—A string that specifies the event name.

Note This string should match the event name that was used to create the Prime Network event using the **vcb event add** command. View the event name using the **vcb event view** command.

Optional Input

subtype—A string that specifies the event subtype name.

Note The subtype parameter is mandatory for INFO events because INFO events do not have subtypes. For events that have subtypes, the subtype parameter is not needed.

Input Format

```
-create_ana_trap_event_type type
-create_ana_trap_event_subtype subtype
```

create-ana-syslog-event

Template to create a unique location ID for a Prime Network syslog event.

Mandatory Input

type—A string that specifies the event name. Mandatory for an INFO event only.

Note This string should match the event name used for creating the Prime Network event using the **vcb event add** command.

Optional Input

subtype—A string that specifies the event subtype name.

Note The subtype parameter is mandatory for INFO events because INFO events do not have subtypes. For events that have subtypes, the subtype parameter is not needed.

Input Format

```
-create_ana_syslog_event_type type
-create_ana_syslog_event_subtype subtype
```

Note For more information about **vcb eventparsingrules** commands, see VCB CLI Reference: vcb eventparsingrules Commands, page B-40.

APPENDIX C

Command Manager and Command Builder: Macro Language and Beanshell Reference

Prime Network Command Manager and Command Builder support two languages for writing command scripts:

- Prime Network Macro Language—Provides simple sequences of Telnet commands, runtime-replaced user-defined input parameters, and inline execution directives that are executed sequentially as Telnet configuration commands on an NE. See Prime Network Macro Language, page C-1.
- BeanShell—Provides a fully programmatic logic via scripting language (including conditions, loops, and external files). See BeanShell Commands, page C-11.

Prime Network Macro Language

These topics describe the Prime Network Macro Language and its syntax, how to use parameters and pragmas, and a detailed example for writing Prime Network Macro Language scripts.

Topics include:

- What Are Prime Network Macro Language Scripts?, page C-1
- Properties Available from the IMO Context, page C-2
- Specifying and Using Parameters, page C-2
- Supported Pragmas, page C-4
- Example, page C-8

What Are Prime Network Macro Language Scripts?

A Prime Network Macro Language script is a simple sequence of Telnet commands, runtime-replaced input arguments, and inline execution directives that are executed sequentially as Telnet configuration commands on a networking device. Prime Network Macro Language script lines are evaluated in runtime for argument replacements that result in the generation of a Telnet device configuration command that can be sent to the device. Each command line is validated according to the inline directives that can abort and roll back the script or continue executing the next script line. Prime Network Macro Language scripts can be created using Command Manager or Command Builder, or can be provided externally using the Prime Network BQL API.

A Prime Network Macro Language script is usually made of a command script and a rollback script. You can specify that if a command script fails, a rollback script is called.

When defining Prime Network Macro Language scripts, you can:

- Import or paste scripts from external sources.
- Define inline directives (pragmas) for validating the network element's reply.
- Define a rollback script for undoing failed commands.

Properties Available from the IMO Context

The script IMO context makes the Prime Network Information Model Objects available to the script as built-in arguments. A script IMO context can be any object that can be represented by a Prime Network IMO, ranging from a managed element to a port connector to a routing entry. Example IMO contexts can include:

NE Type	IMO Name	Example Properties
Managed device	IManagedElement	CommunicationStateEnum, DeviceName, ElementType
Port	IPortConnector	portalias, location, ifindex

For more information about Prime Network Macro Language Built-in parameters, see Built-In Parameters, page C-4.

Specifying and Using Parameters

Prime Network Macro Language supports two types of script parameters: User-defined and built-in; both are replaced at runtime. In the Command Manager and Command Builder GUIs, all parameters (both built-in and user-defined) are available during command editing via a selection list.

Note To view all user-defined and built-in parameters in the Command Manager and Command Builder applications, press **Ctrl-Spacebar** to open the selection list of available arguments (containing both the user-defined input argument and the built-in properties of the IMO context).

Prime Network Macro Language represents both types of parameters in script lines within dollar signs; for example, $...$. For instance, in a VRF configuration command, the input variable vrfName can be defined as ip vrf $vrfName$.

Note

- Timeouts for pragmas and scripts are supported using BQL. This adds a timeout type integer defined in milliseconds. We recommend that if you change the timeout for the pragma, you also change the timeout for the script.
- An example of a timeout for a pragma is `route-target both $rt$ [timeout=2000]`.
- An example of a timeout for a script is `<Timeout type="Integer">5000</Timeout>`.

User-Defined Parameters

User-defined input parameters must be defined up front. A parameter specification includes parameter name, type, and even an optional default value. User-defined parameters can be defined using Command Manager or Command Builder, or through the Prime Network API.

Table C-1 provides a complete list of user-defined parameter properties.

Table C-1 Available User-Defined Parameters

Property	Explanation
Name	Parameter name. Can contain only letters, digits, hyphen (-), and underscore (_), and must be unique.
Caption	Parameter display name. Visible in the Command Manager and Command Builder script execution window.
Type	String, Integer, IPSubnet, Combo, IP, Float, Long.
Width	Field width, in characters. Relevant for the Command Manager and Command Builder script execution window.
Visible	Indicates whether or not the parameter appears in the window. Relevant for the Command Manager and Command Builder script execution window.
Tooltip	Tooltip for the command parameter. **Note** This property is only available through the Command Manager and Command Builder GUIs.
Default	A default value for the parameter. **Note** This property is only available through the Command Manager and Command Builder GUIs.
Required	Indicates whether the argument is mandatory or optional. **Note** This property is only available through the Command Manager and Command Builder GUIs.

Note Some parameter properties are relevant only for the script data entry window in Command Manager and Command Builder.

During runtime, the script is executed via a BQL command. As with all BQL commands, if the argument types do not match, an exception is returned to the user.

User-defined parameters values can be provided in the following ways:

- Using flow-through activation—The input parameters are provided as part of the API before they are sent to the VNE.
- Run from Prime Network Vision as a GUI-based command—You provide the input parameters before they are sent to the VNE; for example, by entering a value or choosing one from a drop-down list.

Multiple Formats for IP Subnet Parameters

Prime Network Macro Language scripts support multiple formats for IP subnet parameters, as described in Table C-2, using the example 198.168.2.10 255.255.255.0.

Table C-2 Formats for IP Subnet Parameters

#	Format	Description	Output
1	maskbits	The IP of the subnet converted to an integer value. Bits only.	30
2	ip	Only the IP without the mask.	198.168.2.10
3	mask	The IP of the subnet mask without the IP address.	255.255.255.0
4	networkmask	The mask address converted to the network.	0.0.0.255
5	ipmaskbits	The IP and the value of the mask bits.	IP/30
6	ipmask	The IP mask. This is the default.	198.168.2.10 255.255.255.0
7	ipmasknot	The IP and the network address.	198.168.2.10 + 0.0.0.255

For example, routeadd$SB:IP$mask$SB:mask$ extracts the IP and then the subnet.

Built-In Parameters

Built-in parameters are the built-in properties available in IMO arguments of the IMO context (such as portalias or status), which are automatically set to their runtime value during execution. The built-in properties include IMO attributes, OID attributes, and instrumentation data.

Note To view all user-defined and built-in parameters in the Command Manager and Command Builder application, press **Ctrl-Spacebar** to open the selection list of available arguments (containing both the user-defined input argument and the built-in properties of the IMO context).

Supported Pragmas

You can insert inline directives (pragmas) in the script lines for increased granularity control. Pragmas are enclosed within square brackets ([…]). Table C-3 lists the pragmas that Prime Network Macro Language scripts support.

Table C-3 Supported Pragmas

Pragma	Short Description	Refer to…
Success	Line-specific success check.	Success
Fail	Line-specific failure check.	Fail
Prompt	Line-specific prompt assertion validation.	Prompt
Rollback	Rollback enable or disable.	Rollback
Activity	Script remarks. These also help determine the failure location.	Activity
Enum	Defining enumerated value substitution.	Enum

Note Wherever the carriage return character is required in the middle of a command line, use the escape sequence **&cr**.

Note You can use multiple pragmas in a single line; when this occurs, all pragmas are analyzed. If the same type of pragma is repeated, only the last one is used.

Success

Description

A *success* pragma is validated against the script line reply. The success pragma verifies that a required substring exists in the reply. If the substring is not found, the script fails.

Syntax

```
[success=<string>]
```

where *<string>* represents the expected return value from the device. *<string>* can be simple text or can contain arguments that are replaced in runtime.

Directives

The pragma succeeds and the script continues only if *<string>* is found in the device reply.

The pragma fails if *<string>* does not exist in the reply.

<string> can be a regular expression; it does not necessarily have to be an exact string to match.

Examples

The following example verifies that the specified VRF $newVrf$ does not already exist:

```
show ip vrf $newVrf$  [success=% No VRF $newVrf$]
```

Using Trial for newVrf, this pragma succeeds if the device reply contains `% No VRF Trial`.

Fail

Description

A *fail* pragma is validated against the script line reply. The fail pragma verifies that a required substring does not exist in the reply.

Syntax

```
[fail=<string>]
```

where *<string>* represents the value that should not be included in the device reply. *<string>* can be simple text or can contain arguments that are replaced in runtime.

Directives

The script fails if *<string>* is found in the device reply. The script continues if *<string>* does not exist in the reply.

<string> can be a regular expression; it does not necessarily have to be an exact string to match.

Example

The following example sets a route distinguisher:

```
rd $newRD$   [fail=% Cannot set RD $newRD$]
```

Using 60:60 for newRD, this pragma yields failure only if the device reply contains `=% Cannot set RD 60:60`.

Prompt

Description

A *prompt* pragma is validated against the next Telnet command prompt. The full prompt pragma verifies that the prompt equals the given string. If the prompt differs from the string, the script fails.

Syntax

```
[prompt=<prompt>]
```

where *<prompt>* represents the new expected prompt. *<prompt>* can be simple text or can contain arguments that are replaced in runtime before being sent to the device.

Directives

The pragma is successful and script execution continues only if the next full prompt equals <prompt>. The pragma fails if the next prompt does not equal <prompt>.

Example

The following example changes the Telnet prompt and validates the change in the newly returned Telnet prompt:

configure terminal [prompt=^router(config)#]

This pragma yields success only if the next device prompt matches router(config)# exactly.

Partial Prompt

Description

A partial *prompt* pragma is validated against the next Telnet command prompt. The partial prompt pragma verifies that the suffix of the prompt equals the given string. If the suffix differs from the string, the script fails.

Syntax

```
[prompt=^<prompt>]
```

where `<prompt>` represents the expected full prompt. `<prompt>` can be simple text or can contain arguments that are replaced in runtime before being sent to the device.

Directives

The pragma is successful and script execution continues only if `<prompt>` is found as the suffix of the device prompt. The pragma fails if `<prompt>` is not found in the suffix of the device prompt.

Example

The following example changes the Telnet prompt and validates the change in the newly returned Telnet prompt:

```
configure terminal [prompt=(config)#]
```

This pragma succeeds only if the next device prompt ends with `(config)#`.

Rollback

Description

A *rollback* pragma determines that rollback will be executed only upon failures from this point onward.

Note Be sure the rollback script restores the device prompt to its original value before the script was initiated.

Directives

If the script fails after the [rollback] marker, then rollback is executed.

Note If the rollback script fails, no additional actions can be performed.

Activity

Description

An *activity* pragma sets the text that, if the script fails, appears in the script's result as the name of the activity that failed. The failed activity name (label) appears in the returned result and in the provisioning event that is generated.

Syntax

```
[activity=<activity>]
```

where *<activity>* represents an inline remark comment. *<activity>* can be simple text or can contain arguments that are replaced in runtime before being sent to the device.

Directives

When a failure occurs later in the script, you are notified of the error by activity name.

Example

```
[activity=now adding the vrf]
```

Enum

Description

An enum pragma defines the values that are used when substituting parameter names into a Telnet string.

Directives

The pragma is successful only if you input one of the values in the list. The pragma fails if you do not input one of the values in the list.

Example

The enum pragma appears at the top of the script:

```
[enum RouteTargetTypeEnum 0=export;1=import]
```

Later in the script, the parameter RouteTargetTypeEnum is used:

```
no route-target $RouteTargetTypeEnum$ $RouteTarget$
```

The value that is substituted into the Telnet command for $RouteTargetTypeEnum$ is export or import instead of 0 or 1.

Example

The following command script and rollback script perform an *Add VRF* configuration. The scripts use user-defined arguments to represent the VRF name, route target, and route distinguisher; several types of pragmas to validate the device reply; and remarks in the command script, and rollback script.

Command Script

```
[enum rd 1=60:60;2=80:80]
show ip vrf $vrfName$ [success=% No VRF named $vrfName$]
[activity=prepare for VRF creation]
config terminal [success=Enter configuration commands, one per line.  End with CNTL/Z.]
[prompt=(config)]
ip vrf $vrfName$ [prompt=(config-vrf)]
[rollback]
[activity=create VRF]
rd $rd$ [fail=% Cannot set RD, check if it's unique]
route-target both $rt$
end
```

Rollback Script

```
config terminal
no ip vrf $vrfName$
end
```

Table C-4 lists the user-defined argument definitions used in the script.

Table C-4 User-Defined Argument Definitions

Name	Type	Default	Explanation	Example
vrfName	String	N/A	The VRF name. The value provided for this argument is used as the VRF table name.	Manhattan
rt	String	N/A	The VRF route target, in the format *integer:integer*. The value provided for this argument is used as is for the device configuration.	60:60
rd	String	1	In this example, the system administrator would like the route distinguisher to be based on the predefined enumerated values list. Therefore, the route distinguisher is provided in the format of an integer to be used as a lookup table key, and not x:y.	1, 2, or any valid value according to the enum pragma

Table C-5 provides an explanation of the command script line by line.

Table C-5 Command Script Explanation

Script Line	Explanation
[enum rd 1=60:60;2=80:80]	The line enumerates the possible values of the route distinguisher argument.
show ip vrf $vrfName$ [success=% No VRF named $vrfName$]	Verify if the requested VRF already exists. Continue to create the VRF only if the requested VRF name is not found.
[activity=prepare for VRF creation]	Remark to state that the following section is preparation for VRF creation.

Table C-5 Command Script Explanation (continued)

Script Line	Explanation
config terminal [success=Enter configuration commands, one per line. End with CNTL/Z.] [prompt=(config)]	Change mode command. Continue to the next command if the success pragma string is found in the device reply and prompt changes to config.
ip vrf $vrfName$ [prompt=(config-vrf)]	Change mode command. Continue to the next command if prompt changes to config-vrf.
[rollback]	Placeholder to state that rollback should be executed only if a subsequent script line fails.
[activity=create VRF]	Remark to state that the following section is actually the VRF creation.
rd rd [fail=% Cannot set RD, check if it's unique]	Set the route distinguisher. If this command fails, the rollback script is called.
route-target both rt	Set the route target. If this command fails, the rollback script is called.
end	Change mode command. Return to normal (enable) mode.

Table C-6 provides an explanation of the activation rollback script line by line.

Table C-6 Rollback Script Explanation

Script Line	Explanation
config terminal	Set the device to terminal mode.
no ip vrf $vrfName$	Delete the VRF from the device.
end	Change mode command. Return to normal (enable) mode.

Running the Script

The script is executed with the following input arguments:

```
vrfName=Trial
rd=2
rt=60:60
```

The Telnet commands as sent to the device (preview):

```
show ip vrf Trial
config terminal
ip vrf Trial
rd 80:80
route-target both 60:60
end
  ------Rollback------
config terminal
no ip vrf Trial
end
```

Full session:

```
vrfName=Trial
rd=2
```

```
rt=60:60

PE-North#show ip vrf Trial
% No VRF named Trial
PE-North#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
PE-North(config)#ip vrf Trial
PE-North(config-vrf)#rd 80:80
PE-North(config-vrf)#route-target both 60:60
PE-North(config-vrf)#end
```

Rerunning the script with the same input values (VRF already exists; the command stops after VRF name verification):

```
PE-North#show ip vrf Trial
  Name                             Default RD          Interfaces
  Trial                            80:80
PE-North#
 ^ Failed to find the text '% No VRF named Trial' in the device reply!, script terminated.
```

Running the script with a different VRF name but the same route target (RT) and route distinguisher (RD) (VRF creation begins and then is rolled back due to RD already in use):

```
vrfName=Trial2
rd=2
rt=50:50

PE-North#show ip vrf Trial2
% No VRF named Trial2
PE-North#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
PE-North(config)#ip vrf Trial2
PE-North(config-vrf)#rd 80:80
% Cannot set RD, check if it's unique
PE-North(config-vrf)#
 ^ Error in activity 'create VRF'.
 ^ Found the text '% Cannot set RD, check if it's unique' in the device reply!, script
terminated.
-----Invoking Rollback-----
PE-North#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
PE-North(config)#no ip vrf Trial2
% IP addresses from all interfaces in VRF Trial2 have been removed
PE-North(config)#end
```

BeanShell Commands

These topics describe the methods that should be used for BeanShell in Prime Network commands when you want to interact with devices:

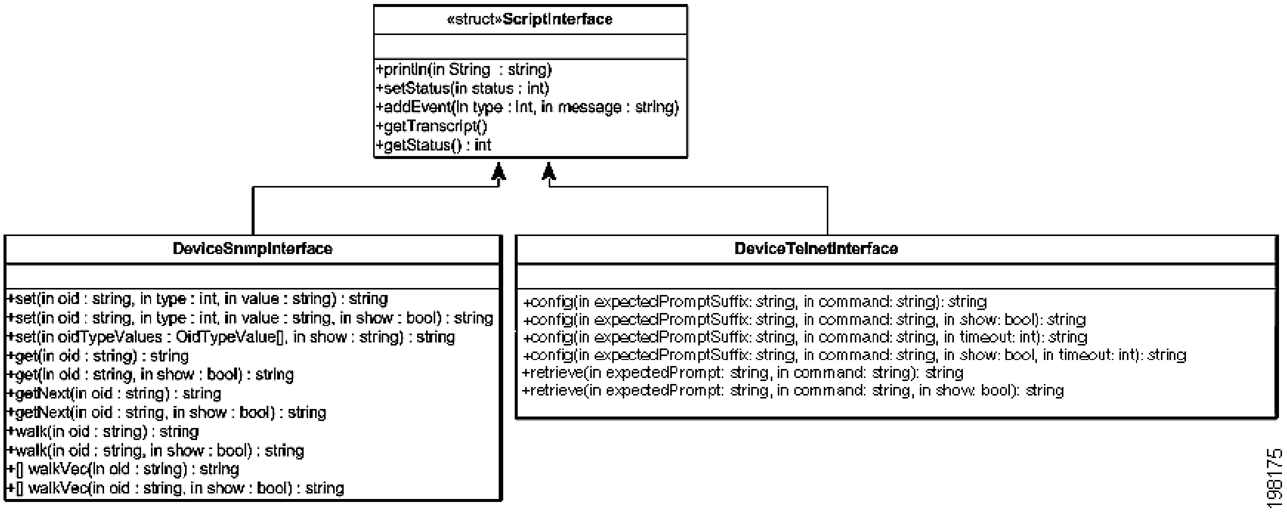
- Telnet BeanShell Commands, page C-12
- SNMP BeanShell Commands, page C-13

. In addition, it provides Telnet and SNMP environment object examples.

Caution Unlike Prime Network Macro Language, in BeanShell user arguments, inventory properties should *not* be embedded within dollar signs ($...$).

Figure C-1 presents the methods that should be used for BeanShell in Prime Network commands when interacting with devices for Telnet and SNMP interfaces.

Figure C-1 BeanShell Methods


«struct»ScriptInterface
+println(in String : string)
+setStatus(in status : int)
+addEvent(In type : Int, in message : string)
+getTranscript()
+getStatus() : int
DeviceSnmpInterface
+set(in oid : string, in type : int, in value : string) : string
+set(in oid : string, in type : int, in value : string, in show : bool) : string
+set(in oidTypeValues : OidTypeValue[], in show : string) : string
+get(in oid : string) : string
+get(In old : string, in show : bool) : string
+getNext(in oid : string) : string
+getNext(in oid : string, in show : bool) : string
+walk(in oid : string) : string
+walk(in oid : string, in show : bool) : string
+[] walkVec(In old : string) : string
+[] walkVec(in oid : string, in show : bool) : string
DeviceTelnetInterface
+config(in expectedPromptSuffix: string, in command: string): string
+config(in expectedPromptSuffix: string, in command: string, in show: bool): string
+config(in expectedPromptSuffix: string, in command: string, in timeout: int): string
+config(in expectedPromptSuffix: string, in command: string, in show: bool, in timeout: int): string
+retrieve(in expectedPrompt: string, in command: string): string
+retrieve(in expectedPrompt: string, in command: string, in show: bool): string
198175

Note For setStatus, 1 = success and 2 = failure.

Telnet BeanShell Commands

The following are examples of the available predefined Telnet environment objects that you can use to interact with a device:

- telnetInterface.config (*prompt*,*telnet_command*, true/false, *timeout*)—Where:
  - *prompt* is the expected prompt after command is executed.
  - *telnet_command* is the actual command to be executed.
  - true displays the results and false hides the results.
  - *timeout* is the CLI time out in milliseconds. The default value is 20000 milliseconds (20 seconds).
- telnetInterface.setStatus (1 or 2)—Where 1 = success and 2 = fail.
- telnetInterface.println—Used to print the output string on screen.

A timeout error reports the failure in the following format:

```
Unexpected error occurred during script execution:
receiveUntil(): general timeout expired(value=<elapsed_time_in_milliseconds>)
(<command_run>)
Elapsed time: <elapsed_time_in_seconds> seconds
```

where:

- *elapsed_time_in_milliseconds* is the length of the timeout in milliseconds.

- *`command_run`* is the command that was being executed when the timeout occurred.
- *`elapsed_time_in_seconds`* is the length of the timeout in seconds.

For example, a timeout error might read as follows:

```
Unexpected error occurred during script execution:
receiveUntil(): general timeout expired(value=10000)(copy tftp://171.69.75.3/radA020C.tmp
null:
Accessing tftp://171.69.75.3/radA020C.tmp...)
Elapsed time: 10 seconds
```

Reload Router Command Example

The following is an example of using BeanShell script to reload a router.

```
try {
  telnetInterface.config("[confirm]", "reload", false);
  telnetInterface.config("#", "\n", false, 2000);
}
catch (Exception e) {
  telnetInterface.println("Router will reload");
}
```

SONET Show Controller Data Command Example

The following is an example of the BeanShell implementation of the SONET Controller Data command that displays SONET controller data:

```
try {
    String sep = File.separator;
    source("." + sep + "scripts" + sep + "configuration" + sep + "cisco" + sep +
      "CiscoUtil.bsh");

    String strOid = oid.toString();
    String SEPARATOR = "PortNumber=POS";

    int startIdx = strOid.indexOf(SEPARATOR);
    startIdx = startIdx + SEPARATOR.length();
    int endIdx = strOid.indexOf(')', startIdx);

    String interfaceName = strOid.substring(startIdx, endIdx);

    telnetInterface.println("Running command: show controller sonet "+interfaceName);
    String res = telnetInterface.retrieve("#", "show controller sonet "+interfaceName);

    telnetInterface.println(res);

    telnetInterface.setStatus(XProvisioningConfigDeviceStatusMsg.STATUS_SUCCESS);

}   catch (Exception e) {
    telnetInterface.setStatus(XProvisioningConfigDeviceStatusMsg.STATUS_FAILURE);
    telnetInterface.println("Exception occurred during execution of the script " +
    e.getMessage() );
}
```

SNMP BeanShell Commands

The following are examples of the available predefined SNMP environment objects that you can use to interact with a device:

- snmpInterface.get (OID, true/false)—Gets OID, and displays or hides results.

- snmpInterface.getNext (OID, true/false)—Gets next OID, and displays or hides results.
- snmpInterface.set (OID, variable type, value)—Sets OID to specified value.
- snmpInterface.walk (OID, true/false)—Returns vector of strings.
- snmpInterface.setStatus (1 or 2)—Where 1 = success and 2 = failure.

For additional information about general scripting language, see http://www.beanshell.org/.

TLI BeanShell Commands

The following is an example of the available predefined TL1 environment object that you can use to interact with a device:

- String config(String command, boolean show) throws Exception;
- String config(String command) throws Exception;
- String retrieve(String command) throws Exception;
- String retrieve(String command, boolean show) throws Exception;
- String waitForEvent(boolean show) throws Exception;
- String waitForEvent() throws Exception;
- String waitForEvent(int timeout) throws Exception;
- String waitForEvent(int timeout, boolean show) throws Exception;

For more information about TL1 commands, see

http://www.cisco.com/en/US/docs/optical/15000r9_1/tl1/sdh/beginners/guide/91e_tlbgn.html#wp44529

TL1 Command to Retrieve General NE Attributes Example

```
import com.sheer.metrocentral.framework.configuration.ScriptInterface;

import com.sheer.metrocentral.framework.provisioning.messages.XProvisioningConfigDeviceStatusMsg;

import com.sheer.metrocentral.framework.configuration.ScriptInterface;

import com.sheer.metrocentral.framework.provisioning.messages.XProvisioningConfigDeviceStatusMsg;

import java.util.HashMap;

import com.sheer.system.os.interfaces.Logger;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.Date;

import com.sheer.util.*;

import java.lang.String;

ScriptInterface protocolInterface = deviceInterface;

String cmd="RTRV-NE-GEN:::123;";

String  result = protocolInterface.send(cmd,true);
```

```
protocolInterface.println(result);
```