Cisco Prime Network 3.8 Customization Guide

November 14, 2011

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

Text Part Number: OL-24647-01
CONTENTS

Preface xv

PART 1 Overview

CHAPTER 1 Prime Network Overview 1-1
   What Is Prime Network? 1-1
   Prime Network Customization Components 1-2
      Soft Properties Manager 1-2
      Prime Network Workflow 1-2
      Command Builder 1-3
      Prime Network Shell Interface 1-3
      Virtual Network Element Customization Builder (VCB) 1-3
      External Launch Application 1-4
      Drools Rules 1-4

PART 2 Soft Properties

CHAPTER 2 Introducing the Prime Network Soft Properties Manager 2-1
   About the Soft Properties Manager 2-1
   Soft Properties 2-2
   Alarm Thresholds 2-2
   Basic Concepts and Terms 2-3

CHAPTER 3 Working with the Soft Properties Manager 3-1
   Launching Soft Properties Manager 3-2
   Soft Properties Manager Window 3-3
      Properties Panel 3-4
      Element Properties Table 3-4
      Soft Properties Manager Menu Bar 3-4
      Soft Properties File Menu 3-5
      Soft Properties Tools Menu 3-5
      Soft Properties Toolbar 3-5
   Soft Properties Manager Workflow 3-6
Creating or Editing a Soft Property 3-7  
Defining the Soft Property General Parameters 3-7  
Defining the Soft Property Parsing Parameters 3-9  
Defining the Parsing Parameters of a Soft Property Table 3-13  
Testing the Parsing Rules 3-14  
Defining TCA Parameters 3-14  
Editing a Soft Property 3-17  
Debugging the Soft Property 3-17  
Viewing the Soft Property in the Inventory Window 3-19  
Publishing the Soft Property 3-19  
Deleting a Soft Property 3-22  
Importing and Exporting a Soft Property 3-22  
Exporting a Soft Property 3-22  
Importing a Soft Property 3-23  

CHAPTER 4  
Soft Property Examples 4-1  
Basic Soft Property Example 4-1  
Soft Property Example Including a TCA 4-8  
Soft Property Table Example 4-9  

CHAPTER 5  
Parsing Operators/Rules 5-1  
Header and Footer 5-1  
Remove Lines 5-3  
Select Lines 5-4  
Replace 5-5  
Match 5-6  
Set 5-7  
Substring 5-8  
Parse Integer 5-9  

CHAPTER 6  
Alarm Threshold Triggers 6-1  
Value Equal 6-1  
Value Not Equal 6-2  
Upper Threshold 6-2  
Lower Threshold 6-2  
Upper Rate 6-3
Lower Rate  6-3

CHAPTER 7

Regular Expressions for Soft Properties Manager  7-1
  Supported Syntax  7-1
  Unsupported Syntax  7-4

PART 3

CHAPTER 8

Prime Network Workflow

Getting Started with Prime Network Workflow  8-1
  About Prime Network Workflow  8-1
  Workflow Concepts  8-2
    Workflow Task  8-2
    BQL Task  8-2
    Lock/Unlock Task  8-3
    Workflow Call (Synchronous Workflow Nesting)  8-3
  Taskflow for Creating a Workflow  8-4

CHAPTER 9

Working with Prime Network Workflow  9-1
  Starting Prime Network Workflow  9-1
  Prime Network Workflow Window  9-3
  Creating a New Workflow Template  9-4
    Workflow Template Names  9-4
    Workflow Attribute Types  9-5
  Retrieving a Workflow Template  9-5
  Deploying a Workflow Template  9-5
  Running a Workflow Template  9-6
  Deleting a Workflow Template  9-6
  Viewing Workflow Properties  9-6
  Working with the Task Library  9-8
    Execute BQL Task  9-9
    Workflow Call Task  9-11
    Subflow Task  9-14
    Lock Task  9-16
    Unlock Task  9-18
  Testing a Workflow  9-20
  Gateway Workflow Commands and Operations  9-22
    Deleting a Template  9-22
Contents

Running a Workflow 9-22
Aborting a Workflow 9-23
Deleting a Workflow 9-23
Getting Workflow Output 9-23

CHAPTER 10  Managing Workflows 10-1
Starting Prime Network Administration 10-1
Viewing the List of Templates and Template Properties 10-2
Deleting a Workflow Template 10-2
Viewing the Output of a Workflow 10-2
Aborting a Workflow 10-3
Deleting a Workflow 10-4
Viewing the Workflow History Using Prime Network Events 10-4

CHAPTER 11  Customizing Prime Network Workflow 11-1
Coding 11-1
Custom Tasks 11-1
Custom Task Panel Factories 11-2
Workflow Callbacks Class 11-2
Packaging for Deployment 11-3
Custom Tasks 11-3
Custom Task Panel Factories 11-3
Workflow Callbacks Class 11-3
Deploying 11-4
About Workflow Migration 11-4

PART 4

CHAPTER 12  Introducing Command Builder 12-1
About Command Builder Migration 12-2

CHAPTER 13  Working with Command Builder 13-1
Opening Command Builder 13-1
Command Builder Window 13-2
Table of Commands 13-2
Menu Options 13-4
Toolbar 13-4
A Workflow to Define a New Command 13-6
Creating a Command 13-7
  Defining a Combo Field Type 13-13
  Defining Prime Network Macro Language Script Lines 13-14
  Defining BeanShell Script Lines 13-16
Managing Commands 13-18
  Executing Commands 13-18
    Executing a Command Immediately 13-18
    Scheduling Command Execution 13-22
    Managing Command Jobs 13-23
  Publishing Commands 13-24
  Editing Commands 13-26
    Editing Predefined Commands 13-27
  Exporting Commands 13-28
  Importing Commands 13-28
  Updating Commands 13-29
  Reviewing Command History 13-30
  Deleting Commands 13-30

CHAPTER 14
Prime Network Macro Language 14-1
  What Are Prime Network Macro Language Scripts? 14-1
  Properties Available from the IMO Context 14-2
  Specifying and Using Parameters 14-2
    User-Defined Parameters 14-3
    Multiple Formats for IP Subnet Parameters 14-4
    Built-In Parameters 14-4
  Supported Pragmas 14-4
    Success 14-5
    Fail 14-6
    Prompt 14-6
    Full Prompt 14-7
    Rollback 14-7
    Activity 14-8
    Enum 14-8
  Example 14-9
    Command Script 14-9
    Rollback Script 14-9
    Running the Script 14-11
Editing a Device Type 18-15
Deleting a Device Type 18-16
Enabling Support for Additional Software Versions 18-16
Viewing the Software Versions 18-16
Adding Support for a New Software Version 18-18
Editing a Software Version 18-18
Deleting a Software Version 18-19
Using the CLI to Create and Manage U-VNEs 18-19
Introductory Command Reference and Global Options 18-20
   vcb 18-20
   vcb sitechanges 18-22
Commands for Adding and Managing U-VNEs 18-23
   vcb uvne add 18-23
   vcb uvne view 18-25
   vcb uvne modify 18-28
   vcb uvne delete 18-30
Commands for Adding and Managing Device Types 18-31
   vcb devicetype add 18-31
   vcb devicetype view 18-33
   vcb devicetype modify 18-34
   vcb devicetype delete 18-35
Testing and Certifying U-VNEs 18-36
Setting Up the Test Environment 18-37
   Creating a Test AVM and VNE 18-38
   Checking the Communication and Investigation State of the Test Instance 18-38
   Measuring Device Performance 18-39
Procedures for Testing and Certifying U-VNEs 18-39
   Preparing Compliance Reports 18-39
   Testing the Physical Inventory 18-40
   Testing the Logical Inventory 18-41
   Testing CPU and Memory Usage 18-42
   Testing That Prime Network Receives Events from an NE and Generates Tickets 18-42
Troubleshooting Device Type Customizations 18-43
Moving a U-VNE to Production 18-44

CHAPTER 19

Enabling Support for Additional Modules 19-1
   Enabling Support for a New Standard Module 19-1
   Adding Modules - Procedure 19-2
   Viewing and Managing Existing Standard Modules 19-3
Contents

Viewing Modules 19-3
Editing Standard Modules 19-4
Deleting Standard Modules 19-5
Enabling Support for a New Pluggable Module 19-5
Managing Existing Pluggable Modules 19-6
Viewing Pluggable Modules 19-7
Editing Pluggable Modules 19-7
Deleting Pluggable Modules 19-8
Testing and Certifying Module Customization 19-9
Using the CLI to Add and Manage Modules 19-9
Planning—Evaluating Module Templates 19-9
Commands for Managing Modules 19-10
  vcb module add 19-11
  vcb module view 19-13
  vcb module modify 19-15
  vcb module delete 19-16
Commands for Managing Pluggable Modules 19-18
  vcb pluggablemodule add 19-19
  vcb pluggablemodule view 19-20
  vcb pluggablemodule modify 19-22
  vcb pluggablemodule delete 19-23

CHAPTER 20
Managing Additional Event Types Using the VCB 20-1
Overview 20-1
Enabling Support for Additional Events 20-2
  Enabling Support for Unsupported Traps 20-2
Adding Support for a Custom Syslog 20-7
  Parsing Rule Group Inheritance Structure 20-10
Customizing Events 20-10
  Changing the Severity of an Event Subtype 20-10
  Dropping an Event 20-11
    Restoring a Dropped Event 20-11
    Restoring a System Default Event 20-11
  Deleting Events 20-12
Testing and Certifying Event Customizations 20-12
  Moving a Tested and Certified Event Customization to Production 20-13
Using the CLI to Customize Events 20-13
  vcb event Commands 20-14
    vcb event add 20-14
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcb event view</td>
<td>20-16</td>
</tr>
<tr>
<td>vcb event modify</td>
<td>20-18</td>
</tr>
<tr>
<td>vcb event delete</td>
<td>20-20</td>
</tr>
<tr>
<td>vcb eventparsingrules add</td>
<td>20-22</td>
</tr>
<tr>
<td>vcb eventparsingrules view</td>
<td>20-24</td>
</tr>
<tr>
<td>vcb eventparsingrules modify</td>
<td>20-26</td>
</tr>
<tr>
<td>vcb eventparsingrules delete</td>
<td>20-27</td>
</tr>
<tr>
<td>vcb eventparsingrules Commands</td>
<td>20-21</td>
</tr>
<tr>
<td>vcb eventpattern add</td>
<td>20-28</td>
</tr>
<tr>
<td>vcb eventpattern view</td>
<td>20-29</td>
</tr>
<tr>
<td>vcb eventpattern modify</td>
<td>20-31</td>
</tr>
<tr>
<td>vcb eventpattern delete</td>
<td>20-32</td>
</tr>
<tr>
<td>vcb eventparsingrules delete</td>
<td>20-27</td>
</tr>
<tr>
<td>Troubleshooting Event Customization (CLI)</td>
<td>20-34</td>
</tr>
<tr>
<td>vcb eventpattern Commands</td>
<td>20-28</td>
</tr>
<tr>
<td>vcb eventpattern add</td>
<td>20-28</td>
</tr>
<tr>
<td>vcb eventpattern view</td>
<td>20-29</td>
</tr>
<tr>
<td>vcb eventpattern modify</td>
<td>20-31</td>
</tr>
<tr>
<td>vcb eventpattern delete</td>
<td>20-32</td>
</tr>
<tr>
<td>vcb eventpattern delete</td>
<td>20-32</td>
</tr>
<tr>
<td>vcb eventparsingrules Commands</td>
<td>20-21</td>
</tr>
<tr>
<td>Troubleshooting Event Customization (CLI)</td>
<td>20-34</td>
</tr>
</tbody>
</table>

**CHAPTER 21**

Deploying and Maintaining Extensions in a Prime Network Production System 21-1

- Viewing VCB Registry Customizations 21-1
- Exporting VCB Registry Customizations 21-3
- Importing VCB Registry Customizations 21-3
- Deleting VCB Registry Customizations 21-3
- Rolling Back VCB Customizations 21-4

**CHAPTER 22**

VCB Template Reference 22-1

- U-VNE Templates 22-1
  - GenericUVNE Template 22-1
    - Generic U-VNE—Physical Inventory Model 22-3
    - Generic U-VNE—Logical Inventory Model 22-3
    - Generic U-VNE—Supported Traps 22-4
    - Generic U-VNE—Supported Events 22-5
    - Generic U-VNE—Limitations 22-5
    - Generic U-VNE—Supported Topologies 22-5
    - Generic U-VNEs—Supported Technologies 22-5
    - Generic U-VNE—Supported Service Events 22-10
- Module Templates 22-10
  - Module Groups and Module Specification Files 22-10
  - Module Templates by Technology 22-14
    - Ethernet (Fixed) 22-14
Contents

Ethernet (Multiloader) 22-15
POS (Fixed) 22-16
POS (Multiloader) 22-17
Channelized T1/E1 (Fixed) 22-18
Channelized OCXX (Fixed) 22-18
ATM (Fixed) 22-19
ATM (Multiloader) 22-19
Multitechnology 22-20
Serial 22-21
ISDN 22-22
Generic 22-22
Ethernet (Cisco Catalyst 3400) 22-23
Ethernet (Cisco Catalyst) 22-23

Event Templates 22-24
Terminology Used in Event Templates 22-24
Supported Interface Types 22-25
Event Templates Functional Summary 22-25
Event Templates Input Summary—Required and Optional Input 22-27
Event Identification Templates 22-29
snmp-trap-identification 22-29
syslog-identification 22-30
Unique ID Templates—for Traps Only 22-31
snmp-trap-identifier-from-oid 22-31
snmp-trap-identifier-from-value 22-32
Event Subtype Templates 22-32
snmp-trap-subtype-from-oid 22-32
snmp-trap-subtype-from-trapoid 22-33
snmp-trap-subtype-from-value 22-33
syslog-subtype-from-expression 22-34
Entity ID Templates—for Traps Only 22-34
snmp-trap-entity-from-oid 22-35
snmp-trap-entity-from-value 22-35
Entity Key Templates 22-35
create-interface-key-from-ifindex 22-36
create-interface-key-from-ifname 22-36
create-managedelement-key 22-36
create-efp-key-from-ifname-serviceid 22-36
create-logical-container-key 22-36
create-moduleDC-key-given-entPhysicalIndex 22-36
create-moduleDC-with-slotSubslot-value-key 22-37
Contents

PART 6

CHAPTER 23

Adding External Launch Points 23-1

About Adding External Launch Points 23-1

Adding an External Launch Point 23-2

Identifying the IMO Context and Properties 23-2

Creating BQL Scripts 23-3

Defining Soft Property Parameters 23-6

Running BQL Scripts 23-7

Enabling Automatic Download of Scripts to Clients 23-7

Accessing the External Launch Point in Prime Network 23-7

Deleting an External Launch Point 23-8

CHAPTER 24

Sample BQL Scripts to Launch External Applications 24-1

PART 7

CHAPTER 25

Drools Rules Management 25-1

About the Drools Rules Engine 25-1

Drools Rules and Prime Network 25-2

Enabling and Disabling Drools Rules 25-3

Drools Rules Definitions in Prime Network 25-3

Modifying a Rule 25-4

Displaying Existing Drools Rules 25-4

Upgrading and Validating Drools Rules Files 25-4

About Drools Rules Migration 25-5

CHAPTER 26

Drools Rules Examples 26-1

Example 1: Generate E-mail Upon Link Down/Up Ticket 26-1

Example 2: Generate E-mail for Critical Ticket 26-2

INDEX
Preface

This guide describes the Cisco Prime Network (Prime Network) 3.8 customization suite and provides instructions for using the components of the customization suite.

This preface includes the following topics:

• Audience, page xv
• Conventions, page xvi
• Related Documentation, page xvii
• Obtaining Documentation and Submitting a Service Request, page xviii

Audience

The following sections describe the primary audience for each tool described in the document.

Soft Properties Manager
Soft Properties Manager is intended for use by integrators and any other users who want to manage the soft properties and Threshold Crossing Alarms (TCAs) that are executed within the Prime Network platform.

Users of Soft Properties Manager are required to have the following preliminary knowledge before using this tool:

<table>
<thead>
<tr>
<th>Subject</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Networking, device properties</td>
<td>Knowledge of device property retrieval using Telnet and SSH or a SNMP MIB browser.</td>
</tr>
<tr>
<td>Data manipulation</td>
<td>Ability to write regular expressions and parse raw strings using basic parsing tools.</td>
</tr>
<tr>
<td>Cisco IMO Types</td>
<td>Understanding of the Prime Network information model so that they know where to locate and edit the soft property.</td>
</tr>
</tbody>
</table>

Command Builder
Command Builder is intended for use by programmers who want to write command scripts that are executed within the Prime Network activation framework.
Workflow
Prime Network Workflow is intended for use by a system administrator.

Prime Network Shell Interface
The Prime Network shell interface is intended for use by system administrators and users who are responsible for network operation tasks.

Drools Rules
The Drools Rules engine is intended for use by system administrators and users who are responsible for troubleshooting the fault in the network.

Conventions
This document uses the following conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bold</strong> font</td>
<td>Commands, keywords, and user-entered text appear in <strong>bold</strong> font.</td>
</tr>
<tr>
<td><em>italic</em> font</td>
<td>Document titles, new or emphasized terms, and arguments for which you supply values are in <em>italic</em> font.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Elements in square brackets are optional.</td>
</tr>
<tr>
<td>{x</td>
<td>y</td>
</tr>
<tr>
<td>[ x</td>
<td>y</td>
</tr>
<tr>
<td>string</td>
<td>A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.</td>
</tr>
<tr>
<td><strong>courier</strong> font</td>
<td>Terminal sessions and information the system displays appear in <strong>courier</strong> font.</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Nonprinting characters such as passwords are in angle brackets.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Default responses to system prompts are in square brackets.</td>
</tr>
<tr>
<td>!, #</td>
<td>An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.</td>
</tr>
</tbody>
</table>

Note
Means reader take note.

Caution
Means reader be careful. In this situation, you might perform an action that could result in equipment damage or loss of data.
Related Documentation

Note

We sometimes update the documentation after original publication. Therefore, you should also review the documentation on Cisco.com for any updates.

The following documentation is available for Prime Network 3.8:

- Cisco Prime Network 3.8 Administrator Guide
- Cisco Prime Network 3.8 User Guide
- Cisco Prime Network 3.8 Documentation Guide
- Cisco Prime Network 3.8 Installation Guide
- Cisco Prime Network 3.8 Quick Start Guide
- Cisco Prime Network 3.8 Reference Guide
- Cisco Prime Network 3.8 Release Notes
- Cisco Prime Network 3.8 User Guide
- Cisco Prime Network 3.8 Change and Configuration Management User and Administration Guide
- Cisco Prime Network 3.8 Activation User Guide
- Cisco Prime Network 3.8 Activation Customization Guide
- Open Source Used in Cisco Prime Network 3.8

Cisco Prime Network 3.8 Integration Developer Guide is available on the Cisco Prime Network Technology Center website. This guide describes how to use Prime Network integration interfaces.

The Cisco Prime Network Technology Center is an online resource for integration developers who use Prime Network application programming interfaces (APIs). It provides information, guidance, and examples to help you integrate your applications with Prime Network. It also provides a platform for you to interact with subject matter experts. To view the information on the Cisco Prime Network Technology Center website, you must have a Cisco.com account with partner level access, or you must be a Prime Network licensee. You can access the Cisco Prime Network Technology Center at http://developer.cisco.com/web/prime-network/home.
Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly What’s New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:


Subscribe to the What’s New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS version 2.0.
PART 1

Overview

This part contains the following chapter:
Chapter 1, “Prime Network Overview”
Prime Network Overview

The following topics introduce you to the Cisco Prime Network (Prime Network) customization products. These topics briefly explain what Prime Network is, and describe the Prime Network customization tools.

- What Is Prime Network?, page 1-1
- Prime Network Customization Components, page 1-2

What Is Prime Network?

Prime Network is a carrier-class network management platform, designed to serve as an active mediation layer between the operation and network layers. It provides a set of easy-to-use applications as well as well defined application programming interfaces (APIs) for Operations Support System (OSS), enabling carriers and service providers to efficiently respond to the constant market demand for new, reliable, and more sophisticated services, while hiding the complexity of large, multivendor, mixed-technology networks.

Prime Network provides solutions for diverse network environments and applications. It offers an integrated network and service autodiscovery for network modeling, intelligent fault analysis, and a highly flexible network configuration and activation engine. This enables fully correlated management of global scale networks supporting millions of subscribers and customers.

Prime Network is a network management solution that provides a fully integrated service-oriented solution offering:

- Multivendor, hybrid device support.
- Mixed technology (IP, Virtual Private Network [VPN], Multiprotocol Label Switching [MPLS], Ethernet, ATM, and Digital Subscriber Line [DSL]).
- Multifunction (network discovery, fault, activation, and configuration).
- Vertical integration with multiple OSS and Business Support Systems (BSS) applications.

Based on a patented, innovative architecture of distributed autonomous virtual network elements (VNEs), Prime Network enables integrated management for hybrid network environments while being scalable to support network growth and evolution. For details, see the Cisco Prime Network 3.8 Administrator Guide.
Prime Network Customization Components

Prime Network customization components provide the ability to monitor and make changes to network elements. These customization components include:

- **Soft Properties Manager**, page 1-2
- **Prime Network Workflow**, page 1-2
- **Command Builder**, page 1-3
- **Prime Network Shell Interface**, page 1-3
- **Drools Rules**, page 1-4
- **Virtual Network Element Customization Builder (VCB)**, page 1-3
- **External Launch Application**, page 1-4
- **Drools Rules**, page 1-4

**Soft Properties Manager**

Using Soft Properties Manager, you can manage soft properties and Threshold Crossing Alarms (TCAs). Soft properties enable you to extend the set of supported properties for each network element (NE), by adding soft properties to the VNEs. 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.

These properties extend the Prime Network model. For more information, see:

- Chapter 2, “Introducing the Prime Network Soft Properties Manager”
- Chapter 3, “Working with the Soft Properties Manager”
- Chapter 4, “Soft Property Examples”
- Chapter 5, “Parsing Operators/Rules”
- Chapter 6, “Alarm Threshold Triggers”
- Chapter 7, “Regular Expressions for Soft Properties Manager”

**Prime Network Workflow**

Using Prime Network Workflow, you can create workflows of tasks, calling multiple activation scripts to perform configuration changes on multiple network elements. You can run activation scripts sequentially, in parallel, on demand, or at a scheduled time. You can view workflows with Prime Network Workflow, Prime Network Administration, and Prime Network Events. Workflows are stored on the gateway, where you can check workflow properties and status. It is intended for use by a system administrator.

Autonomy Workflow Studio can be customized to meet an individual customer’s specific needs. Autonomy Workflow Studio menus and toolbars are customized for use with Prime Network. For more information, see:

- Chapter 8, “Getting Started with Prime Network Workflow”
Command Builder

Using Command Builder (a template-based configuration tool), you can create activation scripts to make physical and logical configuration changes on a network element. Command scripts enable users to execute a programmable sequence of SNMP or Telnet command lines. These scripts can include data properties taken from the Prime Network information model (built-in), as well as user-defined input parameters entered during runtime.

For more information about Command Builder and how to use it, see:

- Chapter 12, “Introducing Command Builder”
- Chapter 13, “Working with Command Builder”
- Chapter 14, “Prime Network Macro Language”
- Chapter 15, “Creating a Prime Network Macro Language Command: An Example”
- Chapter 16, “BeanShell Commands”

Prime Network Shell Interface

The Prime Network shell interface is the command-line interface (CLI) of the Prime Network Shell Manage system. The shell interface provides you with the following features:

- A subset of the system-supported commands so you can manage VNEs, surveillance, and provisioning.
- A flat command hierarchy with a limited number of modes and an unlimited number of nesting levels.
- The ability to export all commands supported by the system.

For more information on the Prime Network shell interface, the supported commands, output formats, and examples, see the Cisco Prime Network 3.8 Integration Developer Guide, available on the Prime Network Technology Center website.

Virtual Network Element Customization Builder (VCB)

The VCB is a tool that allows Prime Network admin users to add Prime Network support for additional NEs, modules, and events, thereby extending what is delivered in Prime Network software. For information, see:

- Chapter 17, “Introducing the VNE Customization Builder (VCB)”
- Chapter 18, “Enabling Support for Additional Device Types and Software Versions.”
- Chapter 19, “Enabling Support for Additional Modules”
- Chapter 20, “Managing Additional Event Types Using the VCB”
- Chapter 22, “VCB Template Reference”
External Launch Application

You can launch an external application from the managed network elements and its attributes from the Prime Network NetworkVision GUI. For more information, see:

- Chapter 23, “Adding External Launch Points.”
- Chapter 24, “Sample BQL Scripts to Launch External Applications.”

Drools Rules

The Drools Rules engine is a general-purpose expert-system generator that combines rule-based techniques and object-oriented programming. The Drools Rules engine enables you to extend the Prime Network alarm correlation mechanism with user-defined rules and business logic. For more information, see:

- Chapter 25, “Drools Rules Management.”
- Chapter 26, “Drools Rules Examples.”
This part contains the following chapters:

- Chapter 2, “Introducing the Prime Network Soft Properties Manager”
- Chapter 3, “Working with the Soft Properties Manager”
- Chapter 4, “Soft Property Examples”
- Chapter 5, “Parsing Operators/Rules”
- Chapter 6, “Alarm Threshold Triggers”
- Chapter 7, “Regular Expressions for Soft Properties Manager”
Introducing the Prime Network Soft Properties Manager

This chapter describes the Prime Network Soft Properties Manager. In addition, it provides a brief explanation of terms used throughout this guide.

Topics include:

- About the Soft Properties Manager, page 2-1
- Soft Properties, page 2-2
- Alarm Thresholds, page 2-2
- Basic Concepts and Terms, page 2-3

Note: Changes to the registry should be performed only with the support of Cisco. For details, contact your Cisco account representative.

Additional information about soft properties is available on the Cisco Developer Network (CDN).

About the Soft Properties Manager

Prime Network provides deep autodiscovery and maintains a live model of the network. This model is based on Cisco’s Device Component Modeling (DCM) architecture, in which each NE is modeled as an interconnected hierarchy of Device Components (DCs), both physical (for example, cards and ports) and logical (for example, forwarding tables and profiles). Each DC maintains a set of properties, which contain its actual data (such as status, configuration, or performance).

When interacting with northbound clients, the DCM information is translated internally into IMOs, Cisco’s TMF513-based Northbound IMO, which is the public language of the Prime Network system with external systems.

The Prime Network property management framework enables you to extend (in runtime) the system’s coverage and capabilities in two areas:

- Soft Properties—Extending the NE data collection and modeling by adding new properties to the DCs, and assigning them to NE MIB variables. The new soft properties are also automatically added to the Northbound IMO.
- Alarm Thresholds—Assigning various types of alarm conditions to soft properties.
All property definitions and parameters are maintained in XML metadata in the registry. To ease the definition process, Prime Network provides a friendly, simple-to-use GUI that guides you through the definition and testing process, and hides the underlying XML definitions.

Soft Properties

By default, Prime Network VNEs model a subset of the device properties, which cover the most important and commonly used properties. Prime Network offers the Soft Properties mechanism to enable user-configurable extension of device modeling, which can cover any unsupported MIB variable. This enables adding new monitored NE properties in runtime to the default set of supported properties.

The Soft Properties mechanism enables quick adaptation to new software upgrades and new requirements that arise during ongoing operation and deployment. It provides the field engineer with the ability to adapt the currently installed Prime Network software to changes in the deployed network.

Every Soft Property is implemented through a set of definitions that determine how to retrieve, parse, and display a certain MIB variable from the NE. The definition process is done through a simple GUI utility, and does not require system restart. Soft properties are retrieved from the NE using SNMP or Telnet/SSH.

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, the Soft Property mechanism enables you to immediately support the new BER feature in the currently installed version.

Alarm Thresholds

The main positioning of Prime Network is as a mediation layer between the network and the operational and business support systems. As such, it abstracts the physical network and provides a generic, vendor-neutral network model, with a consistent information model and interface.

Prime Network also provides you with the ability to leverage its live network model for intelligent data processing within the mediation layer. This ability enables Prime Network to conduct advanced processing in areas such as fault correlation, root-cause analysis, impact analysis, and activation design and validation. It also enables Prime Network to provide processed information to the applications in the upper tiers. Finally, it enables Prime Network to enhance application functionality, while dramatically reducing the application’s complexity and the uploaded data volumes.

Alarm thresholding is one of the major areas in which Prime Network can boost its northbound clients. With this mechanism, Prime Network constantly monitors selected properties and generates an alarm every time these properties cross a user-defined threshold or violate a condition. This eliminates the need for OSS/BSS applications to constantly upload huge amounts of data and process it. Instead, Prime Network filters out irrelevant data, and sends only meaningful notifications.
Basic Concepts and Terms

- Managed Element—Anything managed by the system; usually a component managed by the VNE, such as a device.
- Network Element (NE)—A user-named physical component or device existing in the network.
- Virtual Network Element (VNE)—A virtual representation of a single network element as a modeled component. VNEs all communicate with each other to present Cisco network service management applications with a single, common device abstraction for network element discovery, configuration, status collection, fault analysis, and other basic network (FCAPS) functions. VNEs can be extended to support new application functionality.
CHAPTER 3

Working with the Soft Properties Manager

This chapter describes the Soft Properties Manager working environment and how to access Soft Properties Manager tools. In addition, it describes the process from creating to publishing a soft property.

Topics include:

- Launching Soft Properties Manager, page 3-2
- Soft Properties Manager Window, page 3-3
- Soft Properties Manager Workflow, page 3-6
- Creating or Editing a Soft Property, page 3-7
- Defining the Soft Property General Parameters, page 3-7
- Defining the Soft Property Parsing Parameters, page 3-9
- Testing the Parsing Rules, page 3-14
- Defining TCA Parameters, page 3-14
- Debugging the Soft Property, page 3-17
- Viewing the Soft Property in the Inventory Window, page 3-19
- Publishing the Soft Property, page 3-19
- Deleting a Soft Property, page 3-22
- Importing and Exporting a Soft Property, page 3-22

Note

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

- For their user account, the Configurator default permissions.
- For all assigned device scopes, the Configurator security level.

Creating users with a combination of security access roles lesser than Configurator may have security implications.
Launching Soft Properties Manager

Soft Properties Manager is launched from a specific network element, which could be a managed element or a selected object within a managed element, such as a port. This network element is used to develop and test the soft property. The content that is displayed in the Soft Properties Manager window is based on the location from which it is launched.

Once the soft property has been completed, it can be published and attached to a wider scope of managed elements.

Note
Initially the soft property applies only to the specific object that you are working on during runtime. Once a soft property has been published and the system has been restarted, it is applied to all objects of the same type, according to the location to which it is published.

To open Soft Properties Manager:

**Step 1**
Do one of the following:

- In the Prime Network Vision tree pane or workspace, right-click a managed element to display a shortcut menu. (For more information about the Prime Network Vision window, see the *Cisco Prime Network 3.8 User Guide*.)
- Open the Inventory window for the required managed element, then right-click the required object in the network element; for example, a port or card. (For more information about the Inventory window, see the *Cisco Prime Network 3.8 User Guide*.)

**Step 2**
Choose **Management > Soft Properties Management** from the shortcut menu. The Soft Properties Manager window is displayed.
Soft Properties Manager Window

Figure 3-1 shows an example of the Soft Properties Manager window.

The Soft Properties Manager window displays a table of all existing soft properties according to the selected entity from which it has been launched. In addition, the applicable properties panels for the managed entity from which the Soft Properties Manager was launched are displayed. For example, the properties panels for an ATM port are Location Information, ATM, and DS3.

Note

No soft properties are displayed in the Soft Properties Manager window for a managed element or required object in the network element when it is opened for the first time.

The Soft Properties Manager enables you to:

- Add a new soft property.
- Edit an existing soft property.
- Delete a soft property.
- Import and export soft properties.
- Test a soft property on the selected managed element.
• Publish a soft property and attach it to a wider scope of managed elements.

The Soft Properties Manager consists of the following:

- **Properties Panel, page 3-4**
- **Element Properties Table, page 3-4**
- **Soft Properties Manager Menu Bar, page 3-4**
- **Soft Properties Toolbar, page 3-5**

### Properties Panel

The properties panel is a drop-down list that contains a list of panels equivalent to the panels displayed in Prime Network Vision for the selected network element from which the Soft Properties Manager was launched. For example, if an ATM port is selected as the launching point, the properties panel contains Location Information, ATM, and OC3 properties. You can then select the panel to which to add the property.

### Element Properties Table

The content displayed in the element properties table changes according to the selection made in the properties panel. The following information is displayed in the element properties table of the Soft Properties Manager window:

- **Label**—The name of the property as displayed in the GUI; for example, Port Type. For tables, this is the table name displayed in the tab.
- **Type**—The soft property type, such as Property or Table.
- **Polling**—The polling group specified for the property, such as system or status.
- **Enabled**—Runs (true) or does not run (false) the command.
- **Command Line**—The command execution for this protocol that should be sent to the NE to retrieve the property. This command can be either a Telnet/SSH command or an SNMP GET command for a specific object identifier (OID).

A table can be sorted:

- According to a column by clicking the column heading. The sort icon is displayed next to the selected column heading.
- In ascending or descending order by clicking the column heading. A triangle is displayed next to the selected column heading.

Click a red triangle in a cell to expand the cell and display all information in the cell.

The Location field displays the number of selected rows and the total number of rows in the table, such as 2/16 Selected. In addition, it displays the location of the selected row in the table, such as Line 3.

### Soft Properties Manager Menu Bar

This section provides a description of each option available in the Soft Properties Manager menus:

- **Soft Properties File Menu, page 3-5**
- **Soft Properties Tools Menu, page 3-5**
Soft Properties File Menu

The File menu provides the following options:

- New Element—Create a new soft property. For more information, see Creating or Editing a Soft Property, page 3-7.
- Edit Element—Edit an existing soft property. For more information, see Defining the Soft Property General Parameters, page 3-7.
- Delete Element—Delete a soft property whether or not it has been published. For more information, see Deleting a Soft Property, page 3-22.

Soft Properties Tools Menu

The Tools menu provides the following options:

- Export Element—Save a soft property to a file that can later be imported to another managed element. For more information, see Importing and Exporting a Soft Property, page 3-22.
- Import Element—Copy a soft property from an exported file and import this soft property to another managed element. For more information, see Importing and Exporting a Soft Property, page 3-22.
- Hierarchy Manager—Move the soft property to a different location or change the scope of the soft property across the network hierarchy (publishing). For more information, see Publishing the Soft Property, page 3-19.

Soft Properties Toolbar

Table 3-1 identifies the buttons that appear in Soft Properties Manager.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>📄</td>
<td>New Element</td>
<td>Create a new soft property. For more information, see Creating or Editing a Soft Property, page 3-7.</td>
</tr>
<tr>
<td>📄</td>
<td>Edit Element</td>
<td>Edit an existing soft property. For more information, see Defining the Soft Property General Parameters, page 3-7.</td>
</tr>
<tr>
<td>🗑️</td>
<td>Delete Element</td>
<td>Delete a soft property whether or not it has been published. For more information, see Deleting a Soft Property, page 3-22.</td>
</tr>
<tr>
<td>📧</td>
<td>Export Element</td>
<td>Save a soft property to a file that can later be imported to another managed element. For more information, see Importing and Exporting a Soft Property, page 3-22.</td>
</tr>
<tr>
<td>📧</td>
<td>Import Element</td>
<td>Copy a soft property from an exported file and import this soft property to another managed element. For more information, see Importing and Exporting a Soft Property, page 3-22.</td>
</tr>
<tr>
<td>🌲</td>
<td>Hierarchy Manager</td>
<td>Move the soft property to a different location or change the scope of the soft property across the network hierarchy. For more information, see Publishing the Soft Property, page 3-19.</td>
</tr>
</tbody>
</table>
Soft Properties Manager Workflow

Figure 3-2 illustrates the steps required to define a new soft property definition using Soft Properties Manager and the order in which they must be performed.

For more information, see the following sections:

- Creating or Editing a Soft Property, page 3-7
- Defining the Soft Property General Parameters, page 3-7
- Defining the Soft Property Parsing Parameters, page 3-9
- Testing the Parsing Rules, page 3-14
- Defining TCA Parameters, page 3-14
- Debugging the Soft Property, page 3-17
- Viewing the Soft Property in the Inventory Window, page 3-19
- Publishing the Soft Property, page 3-19
Creating or Editing a Soft Property

Soft Properties Manager enables you to create or edit an existing soft property using the Add Soft Property dialog box. First, you must determine the managed element or selected object in the network element to which the soft property should be added.

To create a soft property:

Step 1
In Prime Network Vision:
- Right-click a managed element in Prime Network Vision or the workspace.
- Open the Inventory window for the required managed element, then right-click the required object.

Step 2
Choose Management > Soft Properties Management. The Soft Properties Manager window is displayed.

Step 3
Open the Add Soft Property dialog box in one of the following ways:
- Click New Element in the toolbar.
- Choose File > New Element.

The Add Soft Property dialog box is divided into the following tabs (configuration categories):
- General—Configure general definitions for the soft property. For more information, see Defining the Soft Property General Parameters, page 3-7.
- Parsing—Configure parsing definitions for the soft property. For more information, see Defining the Soft Property Parsing Parameters, page 3-9.
- TCA Alarms—Configure alarm threshold management for the soft property. For more information, see Defining TCA Parameters, page 3-14.

Defining the Soft Property General Parameters

The General tab enables you to configure general definitions for the soft property. You can also configure just a single soft property field or an entire soft property table.

To define the general parameters:

Step 1
Select the General tab in the Add Soft Property dialog box.

The following fields are displayed:
- Name—The soft property identifier, which is unique to the location and IMO scope. This field is mandatory.

Note
A warning message appears if the name specified already exists. You are asked whether to override the existing soft property implementation.
Creating or Editing a Soft Property

- **Label**—The soft property name that is displayed in the GUI and which is unique to the location and IMO scope. For tables, this is the table name that is displayed in the tab. This field is mandatory.
- **Description**—A description of the soft property.

The following drop-down lists are displayed:
- **Type**—The soft property type, either Property (default value) or Table. The fields displayed in the Parsing tab depend on your selection.

  **Note** If you select Table, the TCA Alarms tab does not appear in the Add Soft Property dialog box.

- **Polling Rate**—The polling rate group to which the soft property is assigned:
  - topo_I2
  - topo_I1
  - configuration
  - status
  - system

The following check box is displayed:
- **Enabled**—Indicates whether the VNE is to run the command. This option is enabled by default.

The following buttons are displayed in the Add Soft Property dialog box:
- **OK**—Validates the changes according to the field’s rules; that is, the definitions defined in the General, Parsing, and TCA Alarms tabs. If validation is successful, the changes are saved and the Add Soft Property dialog box closes.

  **Note** A warning message is displayed if the validation is unsuccessful.

- **Cancel**—Closes the Add Soft Property dialog box without saving any changes.
- **Debug**—Opens the Debug Soft Property dialog box, displaying the status of the soft property debug.

**Step 2** Define the soft property’s general parameters.

  **Note** If you select the soft property type Table, proceed to Defining the Parsing Parameters of a Soft Property Table, page 3-13 to continue creating the soft property.

Proceed to Defining the Soft Property Parsing Parameters, page 3-9 to define the parsing parameters of the soft property.
Defining the Soft Property Parsing Parameters

The Parsing tab enables you to configure, view, and edit parsing definitions defined for the soft property. To define the parsing parameters:

**Step 1** Select the Parsing tab in the Add Soft Property dialog box. (The Parsing tab is available if you select the type Property in the General tab.)

*Note* If you select the type Table in the General tab, only the SNMP option is available. For more information about defining parsing parameters for a soft property table, see Defining the Parsing Parameters of a Soft Property Table, page 3-13.

The Parsing tab enables you to specify either a Telnet/SSH command or a MIB OID for an SNMP GET command.

The following radio buttons are displayed in the Parsing tab:

- Use SNMP get(OID)—The SNMP retrieval expression. The text field is enabled and mandatory when this option is selected. The expression can include environmental arguments.

*Note* When Use SNMP get(OID) is selected, press **Ctrl-Spacebar** in the text box to view a list of all parameters available for all protocols.

Figure 3-3 shows an example of the output if you press Ctrl-Spacebar for a port.

Figure 3-3 Output

Sometimes, when building a soft property, the Telnet/SNMP command is context sensitive. A good example of this is when you want to retrieve port-related data through SNMP. Searching all ports to find the relevant port each time is not efficient and can greatly affect system performance. To solve this problem, *instrumentation data* is available for the soft property. Instrumentation data is a variant between different elements in the system, depending on the context object you want to add the soft property to.

In this example, the instrumentation data is the port ifIndex. To use the ifIndex in the OID, use 1.3.1.6......$ifIndex$.5.6.4. To determine the instrumentation data that is available for your context object, press **Ctrl-Spacebar** while the cursor is in the command field.
• Use Telnet/SSH—The Telnet/SSH retrieval expression. The text field is enabled and mandatory when this option is selected. The expression can include environmental.

**Note** When Use Telnet/SSH is selected, press **Ctrl-Spacebar** in the text box to open a list of all parameters available for all protocols.

The text area in the Parsing tab enables you to enter the command line of the protocol. For example, for SNMP enter the OID of the SNMP packet, and for Telnet enter the Telnet command line.

**Note** The SNMP OID must start with a dot. For example, if you want to retrieve the OID value of 1.3.6…. you must enter .1.3.6…..

The following columns are displayed in the table of the Parsing tab:

• Index—Displays the order of the parsing rules.

• Operation—Displays the parsing operator type selected in the Add/Edit Parsing Rule dialog box. For more information about parsing operators, see Chapter 5, “Parsing Operators/Rules.”

The following buttons are displayed in the Parsing tab:

• Add—Add a new operator. The Add/Edit Parsing Rule dialog box is displayed.

• Edit—Edit an existing operator. The Add/Edit Parsing Rule dialog box is displayed.

**Note** You can also edit an operator by double-clicking the required operator in the table.

• Delete—Delete the selected operator from the table.

**Note** Delete an operator by selecting it in the table, then clicking **Delete**.

• Test—Enables you to test the soft property parsing. The Test Parsing Rules dialog box is displayed. For more information, see Testing the Parsing Rules, page 3-14.

The reorder arrows enable you to move the rules up and down within the sequence to change the parsing order.

**Step 2** Select the type of retrieval expression: Use SNMP get(OID) or Use Telnet/SSH.
Step 3  Click Add. The Add/Edit Parsing Rule dialog box is displayed (see Figure 3-4).

Figure 3-4  Add/Edit Parsing Rule Dialog Box

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Argument 1</td>
</tr>
<tr>
<td>2</td>
<td>Argument n</td>
</tr>
<tr>
<td>3</td>
<td>Source</td>
</tr>
<tr>
<td>4</td>
<td>Result</td>
</tr>
</tbody>
</table>

The Add/Edit Parsing Rule dialog box is used for adding, editing, and testing parsing operators. Once the required operator is selected, the corresponding arguments are displayed.

The following drop-down lists are displayed in the Add/Edit Parsing Rule dialog box:

- **Input Buffer**—Displays the list of output arguments defined in previous operators and the default (the standard output buffer of the last predecessor operator that was not redirected into an output argument). This field is mandatory.

The parsing result of operator N is available by default as input for operator N+1 (it appears as Default in the Input Buffer field). The parsing result of operator N might be directed to a locally defined environment argument. In this case, the input for operator N+1 is the same as for operator N. Changing the default input buffer is supported by selecting an input buffer other than Default. The available input buffers for Operator N+1 are the set of output arguments defined in Operators 1 through N.
Creating or Editing a Soft Property

The fields displayed in the Expression area of the window change depending on your selection in the Operator drop-down list, which is mandatory. When an operator is selected, the corresponding arguments are displayed. The Operator drop-down list enables you to select one of the available parsing operators:

- **Header And Footer**—Removes a specified number of lines from the header and footer of the input text. For more information, see Header and Footer, page 5-1.
- **Remove Lines**—Removes a range of lines from the specified starting row to the specified end row of the input text. For more information, see Remove Lines, page 5-3.
- **Select Lines**—Extracts a range of lines from the specified starting row to the specified ending row of the input text. For more information, see Select Lines, page 5-4.
- **Replace**—Finds one substring or all occurrences of a substring, which matches a specified regular expression, and replaces it with a specified value. For more information, see Replace, page 5-5.
- **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. For more information, see Match, page 5-6.
- **Set**—_formats the input buffer and local arguments defined in previous operators using a regular expression. For more information, see Set, page 5-7.
- **Substring**—Extracts a substring of a specified length from a specified starting point. For more information, see Substring, page 5-8.
- **Parse Integer**—Uses the substring rule and, when a result is received with the substring, converts it into an integer value. For more information, see Parse Integer, page 5-9.

**Note** If the substring operator contains any characters, the parsing integer operator fails.

The Argument 1 and Argument n fields displayed in the Add/Edit Parsing Rule dialog box are dynamic and list arguments according to the selected operator. For more information about operators and corresponding arguments, see Chapter 5, “Parsing Operators/Rules.”

The Direct result to output buffer variable check box and corresponding field enable you to direct the parsing output to the provided argument instead of it being the input value for the next operator. The text box is enabled only when the check box is selected. A unique (within the complete parsing sequence of this soft property instance) argument name must be provided in this field.

The Simulation area is divided into two text areas:

- The Source text box is used to enter or paste parsing information.
- The Result text box is used to view the parsed result.

The information entered and displayed in the Simulation area is not saved when the dialog box is closed. The Test Expression button parses the information entered in the Source text box according to the parsing operator defined and displays the result in the Result text box.

The following buttons are displayed in the Add/Edit Parsing Rule dialog box:

- **OK**—Validates the operator according to the selected operator’s validation rules. Saves the changes and closes the Add/Edit Parsing Rule dialog box.
- **Cancel**—Closes the Add/Edit Parsing Rule dialog box without saving the changes.

**Step 4** Define the parsing rules.
Proceed to Testing the Parsing Rules, page 3-14 to test the parsing rules.

Defining the Parsing Parameters of a Soft Property Table

The fields that appear in the Parsing tab depend on your selection in the Type drop-down list in the General tab. For example, if you select Table in the Type drop-down list in the General tab, the Telnet/SSH options do not appear in the Parsing tab.

To define the Parsing parameters of a soft property table:

---

**Step 1** Select the Parsing tab in the Add Soft Property dialog box.

The following radio button is displayed in the Parsing tab:

- **Use SNMP get(OID)**—The SNMP retrieval expression. The text field is mandatory when this option is selected. The expression can include environmental arguments.

  **Note** When Use SNMP get(OID) is selected, press **Ctrl-Spacebar** in the text box to view a list of all parameters available for all protocols.

The following columns are displayed in the table of the Parsing tab:

- **Title**—Displays the column headings of the table.
- **OID**—Displays the column information for the table.

  For more information about the buttons displayed in this tab, see Defining the Soft Property Parsing Parameters, page 3-9.

**Step 2** Click Add. The Add/Edit Column Controller dialog box is displayed.

The Add/Edit Column Controller dialog box enables you to add or edit the columns and information displayed in the table.

The following fields are displayed in the Add/Edit Column Controller dialog box:

- **Column Title**—The column heading displayed in the table.
- **Column Data**—The column information displayed in the table.

The following buttons are displayed in the Add/Edit Column Controller dialog box:

- **OK**—Saves the column information and closes the Add/Edit Column Controller dialog box.
- **Cancel**—Closes the Add/Edit Column Controller dialog box without saving any changes.
- **Apply**—Saves the column information; the Add/Edit Column Controller dialog box remains open.

**Step 3** Define the soft property’s table information.

**Step 4** Click **OK**. The Parsing tab is displayed along with the newly defined table information.

Proceed to Debugging the Soft Property, page 3-17 to debug the soft property table.
Testing the Parsing Rules

The Soft Properties Manager provides testing functions, which enable you to simulate each parsing rule as well as the whole parsing sequence. In addition, you can test and debug the property on the NE. You can test and simulate the actual input parsing according to the parsing operators and display the parsing result. This tests all defined parsing rules in the order in which they are given.

To test the parsing rules:

Step 1 Click Test Expression in the Add/Edit Parsing Rule dialog box. The Test Parsing Rules dialog box is displayed.

The Input Buffer area enables you to enter the input to parse. You can copy and paste the parsing input or it can be retrieved from the device. The Input Buffer area is mandatory.

The Output Buffer area displays the parsing result log. It can contain the final parsing result or the entire parsing log with comments for each parser used.

The following buttons are displayed in the Test Parsing Rules dialog box:

- Close—Closes the Test Parsing Rules dialog box.
- Test—Parses the input entered in the Input Buffer area according to the parsing operators. The parsing result is displayed in the Output Buffer area.

Step 2 Click Test. The results of the test are displayed Test Parsing Rules dialog box.

Step 3 Click Close. The Add/Edit Parsing Rule dialog box is displayed.

Step 4 Click OK. The Parsing tab is displayed.

Proceed to Defining TCA Parameters, page 3-14 to define the TCA parameters for the soft property.

Defining TCA Parameters

The TCA Alarms tab enables you to set threshold conditions for the soft property value, which generates an alarm when crossed.

This feature allows you to:

- Select the severity level to associate with an alarm.
- Enable or disable an alarm.
- Select the threshold type.

You can define multiple alarms for the same soft property. The alarm is displayed in the ticket pane of the Prime Network Vision window.
To define TCA parameters:

**Step 1**
Select the **TCA Alarms** tab.

**Note** The TCA Alarms tab is displayed only if you select the type Property in the General tab.

The following columns are displayed in the TCA Alarms tab:
- **Description**—A free text area used to describe the alarm.
- **Trigger**—Displays the details of the trigger selected; that is, what causes the alarm to be sent. For more information, see Chapter 6, “Alarm Threshold Triggers.”
- **Enabled**—Displays the status of the alarm: enabled (true) or disabled (false), as defined in the General tab of the Add TCA dialog box.

The following buttons are displayed in the TCA Alarms tab:
- **Add**—Opens the Add TCA dialog box, enabling you to define the parameters of the TCA.
- **Edit**—Opens the Add TCA dialog box, enabling you to edit the parameters of a previously defined TCA.

**Note** Edit the parameters of a TCA by selecting it in the table, then clicking **Edit**.

- **Delete**—Deletes the selected TCA from the TCA Alarms tab.

**Note** Delete a TCA by selecting it in the table, then clicking **Delete**.

**Step 2**
Click **Add**. The Add TCA dialog box is displayed. It is divided into the following tabs:
- **General**—Define the general parameters, such as the severity and name, of the TCA.
- **Trigger**—Define the alarm threshold trigger for the TCA. For more information, see Chapter 6, “Alarm Threshold Triggers.”

The following fields are displayed in the General tab of the Add TCA dialog box:
- **Name**—The alarm name that is displayed in the ticket pane when the alarm is triggered.
- **Description**—A description of the alarm.

The following check boxes are displayed in the General tab of the Add TCA dialog box:
- **Enabled**—Enables or disables the alarm; select this option to enable the alarm.
- **Can be correlated to other alarms**—Select this option to correlate this alarm to other alarms. For more information about correlating alarms, see the *Cisco Prime Network 3.8 User Guide*.
- **Other alarms can correlate to this alarm**—Select this option to enable other alarms to be correlated to this alarm. For more information about correlating alarms, see the *Cisco Prime Network 3.8 User Guide*. 
Defining TCA Parameters

The following drop-down list is displayed in the General tab of the Add TCA dialog box:

- **Alarm Severity**—Select the severity level associated with the alarm:
  - Critical
  - Major
  - Minor
  - Warning
  - Normal

For more information about alarm severity, see the *Cisco Prime Network 3.8 User Guide*.

The following buttons are displayed in the General tab of the Add TCA dialog box:

- **OK**—Saves the parameters of the TCA and closes the Add TCA dialog box.
- **Cancel**—Closes the Add TCA dialog box without saving any changes.

**Step 3** Define the General parameters for the TCA.

**Step 4** Select the **Trigger** tab and define the Trigger parameters for the TCA:

In the Trigger drop-down list, select a threshold type:

- **Value Equal**—The alarm condition is reached when the soft property value is equal to the value defined in Alarm Value, regardless of whether or not it is numeric. For more information, see *Value Equal*, page 6-1.

- **Value Not Equal**—The alarm condition is reached when the soft property value is not equal to the value defined in the Alarm Value, regardless of whether or not it is numeric. For more information, see *Value Not Equal*, page 6-2.

- **Upper Threshold**—The upper threshold value which, when crossed, triggers the alarm for the defined numeric properties. For more information, see *Upper Threshold*, page 6-2.

- **Lower Threshold**—The lower threshold value which, when crossed, triggers the alarm for the defined numeric properties. For more information, see *Lower Threshold*, page 6-2.

- **Upper Rate**—The upper rate threshold value for the performance counters which, when crossed, triggers the alarm for the defined numeric properties. For more information, see *Upper Rate*, page 6-3.

- **Lower Rate**—The lower rate threshold value for the performance counters which, when crossed, triggers the alarm for the defined numeric properties. For more information, see *Lower Rate*, page 6-3.

For more information about triggers, see Chapter 6, “Alarm Threshold Triggers.”

If you do not want the alarm to be triggered immediately, select the check box and specify the number of seconds the trigger condition should persist until the alarm is triggered.

**Step 5** Click **OK**. The TCA Alarms tab is displayed.

Proceed to *Debugging the Soft Property*, page 3-17 to debug the soft property.
Chapter 3      Working with the Soft Properties Manager

Editing a Soft Property

You can edit an existing soft property. The soft property that is edited affects only the local instance. When an inherited soft property is edited, the new local instance overrides the generic soft property definition for the specific managed element.

To edit a soft property:

---

**Step 1** In the Soft Properties Manager window, select the soft property that you want to edit.

**Step 2** Open the hierarchy manager table in one of the following ways:
- Click **Edit Element** in the toolbar.
- Choose **File > Edit Element**.
- Right-click the soft property, then choose **Edit Element**.

The hierarchy manager table is displayed.

---

**Note** If user-friendly VNE names exist in the schema, the hierarchy manager table displays the user-friendly registry location names in the VNE Hierarchy Location column. A user-friendly VNE name is a hierarchy path that has been defined in the registry and is then displayed in the hierarchy manager table. For more information, see **Publishing the Soft Property**, page 3-19.

---

**Step 3** Select the required version of the soft property from the hierarchy manager, then click **OK**. The Edit Soft Property dialog box is displayed for the selected soft property.

**Step 4** Edit the soft property as required by changing the existing information using the Soft Properties Manager window.

**Step 5** When the soft property has been successfully edited, click **OK** in the Edit Soft Property dialog box to save your changes. The edited soft property is supported and appears in the Soft Properties Manager window.

---

For information about viewing edited soft properties, see **Viewing the Soft Property in the Inventory Window**, page 3-19. For information about publishing edited soft properties, see **Publishing the Soft Property**, page 3-19.

Debugging the Soft Property

You can debug the soft property on the managed element or selected object in the network element by opening the Debug Soft Property dialog box, which displays the status of the soft property when it is debugged. For example, you might want to confirm that a selected device supports Telnet. (See Chapter 4, “Soft Property Examples” for more information.)

---

**Note** No notification message is displayed when a VNE fails to collect properties. The properties are not populated in the GUI.
To debug the soft property:

**Step 1** Click **Debug** in the Add Soft Property dialog box. The Debug Soft Property dialog box is displayed with the results of the debug.

**Figure 3-5 Debug Soft Property Dialog Box**

![Debug Soft Property Dialog Box]

**Step 2** Click **Close**. The Add Soft Property dialog box reappears.

**Step 3** Click **OK**. The Soft Properties Manager window is displayed with the newly created soft property displayed in the element properties table.

**Step 4** Click **Close**. The Prime Network Vision or Inventory window is displayed, depending on your original selection.

Proceed to Viewing the Soft Property in the Inventory Window, page 3-19 to view the results of the soft property in the Inventory window.
Viewing the Soft Property in the Inventory Window

After creating or editing a soft property, you can view the results in the Properties pane of the Inventory window for the managed element or selected object in the network element.

**Note**
You can view the soft property in the Inventory window only after it has been closed and reopened. For example, if you open the Inventory window for a VNE and add a property to one of the ports, it is displayed only after you close and reopen the Inventory window for the VNE.

To view the soft property:

**Step 1** Right-click the required managed element in the tree pane or workspace of the Prime Network Vision window, then select **Inventory**. The Inventory window for the required managed element is displayed with the newly defined soft property or soft property table.

**Step 2** Click the top right corner to close the Inventory window.

Proceed to **Publishing the Soft Property, page 3-19** to publish the soft property.

Publishing the Soft Property

A property definition is applicable to all objects of the same type in the selected NE. However, you might want to apply the same property definitions to all NEs of the same type or family. This requires moving the property definition from the specific NE instance to a higher level in the registry hierarchy.

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 elements in the network.

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**
User with a Configurator role can add and publish soft properties on all devices regardless of the device scope defined for the user.

Different variations of a soft property can be used for different managed elements and network elements, where the implementation of the soft property is different for each managed element or network element.
Figure 3-6 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.

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.

**Note**
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. For more information about measuring the effect of publishing the soft property, contact your Cisco account representative.

**Note**
Soft property publishing sometimes deeply affects system memory usage, device utilization, and system performance. For example, you can add a property that contains the `show running-config` command to a device and publish it to a group of devices. This is likely to cause both system memory usage and device utilization to jump.

To publish a soft property:

**Step 1** In Prime Network Vision, do one of the following:
- Right-click a managed element in the tree pane or workspace.
- Open the Inventory window for the required managed element, then right-click the required object.

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

**Step 3** Select the required properties panel and soft property in the element properties table.

**Step 4** Open the hierarchy manager table in one of the following ways:
- Click Hierarchy Manager in the toolbar.
- Choose Tools > Hierarchy Manager.
- Right-click the item, then choose 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 3-2 describes the icons that appear.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Copy icon]</td>
<td>Copy</td>
<td>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.</td>
</tr>
<tr>
<td>![Cut icon]</td>
<td>Cut</td>
<td>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.</td>
</tr>
<tr>
<td>![Paste icon]</td>
<td>Paste</td>
<td>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.</td>
</tr>
<tr>
<td>![Delete icon]</td>
<td>Delete</td>
<td>Deletes the soft property from the selected node in the hierarchy.</td>
</tr>
</tbody>
</table>

The following button is displayed:

- **Close**—Closes the Hierarchy Manager window without publishing the soft property.

**Step 5**  Select the node in the hierarchy from which you want to publish the soft property.

**Step 6**  Click **Copy** or **Cut** in the toolbar to copy or cut the soft property.

**Step 7**  Select the node in the hierarchy where you want to publish the soft property.

**Step 8**  Click **Paste** on the toolbar to paste the soft property.

**Step 9**  Restart the VNE on which you want to publishing to take effect.
Deleting a Soft Property

Soft properties that you create are, by default, always created as a local instance. A soft property that is defined locally is selected in the Soft Properties Publish Controller dialog box. You can delete soft properties whether or not they have been published.

To delete a soft property:

Step 1  Select the soft property that you want to delete in the element properties table of the Soft Properties Manager dialog box.

Step 2  Delete the soft property in one of the following ways:

- Click **Delete Element** in the toolbar.
- Choose **File > Delete Element**.
- Right-click the soft property, then choose **Delete Element**.

A warning message is displayed.

Step 3  Click **Yes**. The soft property is deleted and no longer appears in the element properties table of the Soft Properties Manager window.

Importing and Exporting a Soft Property

The Soft Properties Manager enables you to export (save) a soft property definition to a file. The soft property definition can then be imported (copied) to another managed element.

In addition, you can export and import a soft property definition to a file and publish it to multiple places in the Hierarchy Manager window.

Exporting a Soft Property

To export a soft property:

Step 1  Select the soft property that you want to export in the element properties table of the Soft Properties Manager dialog box.

Step 2  Open the Export dialog box in one of the following ways:

- Click **Export Element** in the toolbar.
- Choose **Tools > Export Element**.
- Right-click the property, then choose **Export Element**.

The Export dialog box is displayed.

**Note**  If user-friendly VNE names exist in the schema, the hierarchy manager table displays these user-friendly registry location names in the VNE Hierarchy Location column. A user-friendly VNE name is a hierarchy path that has been defined in the registry and is then displayed in the hierarchy manager table. For more information, see **Publishing the Soft Property, page 3-19**.
Step 3  Select the version that you want to export in the table of the Export window. The version is selected in the table.

Step 4  Click OK. The Export Property dialog box is displayed.

Step 5  Browse to the directory where you want to save the soft property.

Step 6  In the File name field, enter a name and extension (for example, .txt) for the soft property.

Step 7  Click Save. The soft property is saved in the selected directory. The Export dialog box is displayed.

Step 8  Click Close. The Soft Properties Manager window is displayed.

Importing a Soft Property

To import a soft property:

Step 1  Open the Import Element dialog box in one of the following ways:
  •  Click Import Element in the toolbar.
  •  Choose Tools > Import Element.

Step 2  Browse to the directory and soft property that you want to import.

Step 3  Click Open. The Import Elements window is displayed.

Step 4  Select the version that you want to import in the table of the Import Elements window. The version is selected in the table.

Step 5  Click OK. The Soft Properties Manager window is displayed.

Step 6  Click Close. The soft property is imported to the selected managed element or network element and displayed in the Soft Properties Manager window.
Soft Property Examples

This chapter provides several examples that describe how to create a soft property from start to finish, including defining TCAs and a soft property table. Topics include:

- Basic Soft Property Example, page 4-1
- Soft Property Example Including a TCA, page 4-8
- Soft Property Table Example, page 4-9

Basic Soft Property Example

To create a basic soft property (excluding a TCA):

Step 1 Do one of the following:
- Right-click a managed element in Prime Network Vision.
- In Prime Network Vision, open the Inventory window for the required managed element, then right-click the required object.

Step 2 Choose Management > Soft Properties Management. The Soft Properties Manager window is displayed.

Step 3 Select the required property from the drop-down list in the properties panel.

Step 4 Open the Add Soft Property dialog box in one of the following ways:
- Click New Element on the toolbar.
- Choose File > New Element.

Step 5 Define the soft property information in the General tab as follows:
- Name—sp01
- Label—My Soft Property
- Description—Example of soft property
- Type—Property
- Polling Rate—Status
- Enabled—Selected

Step 6 Select the Parsing tab.
**Step 7** Define the information in the Parsing tab as follows:
- Select **Use Telnet/SSH**.
- In the text box, enter `show ip vrf example`.

**Step 8** Click **Add**. The Add/Edit Parsing Rule dialog box is displayed (see **Figure 4-1**).

*Figure 4-1 Add/Edit Parsing Rule Dialog Box*

**Step 9** Define the information in the Add/Edit Parsing Rule dialog box as follows:
- **Operator**—Match
- **Expression**—\d\d
- **Source text box**—Enter the information as shown in the example.

**Step 10** Click **Test Expression**. The result 55 is displayed in the Result text box.
**Step 11** Click OK. The Parsing tab of the Add Soft Property dialog box is displayed.

*Figure 4-2*  
**Add Soft Property Dialog Box**

**Step 12** Click Add. The Add/Edit Parsing Rule dialog box is displayed.

*Figure 4-3*  
**Add/Edit Parsing Rule Dialog Box**
Step 13  Define the information in the Add/Edit Parsing Rule dialog box as follows:

- Operator—Substring
- From Index—1
- To Index—1
- Source text box—55

Step 14  Click Test Expression. The result 5 is displayed in the Result text box.

Step 15  Click OK. The Parsing tab of the Add Soft Property dialog box is displayed.

Figure 4-4  Parsing Tab
**Step 16**  Click **Test**. The Test Parsing Rules dialog box is displayed, enabling you to test all defined parsing rules in the order given.

**Figure 4-5  Test Parsing Rules Dialog Box**

**Step 17**  Click **Test**. The result of the test is displayed in the Output Buffer area.

**Step 18**  Click **Close**. The Add Soft Property dialog box is displayed.
Step 19  Click **Debug**. The Debug Soft Property dialog box is displayed.

![Debug Soft Property Dialog Box](image)

**Figure 4-6  Debug Soft Property Dialog Box**

Step 20  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 21  Click **Close**. The Add Soft Property dialog box is displayed.

(To add a TCA, go to Step 2 in **Soft Property Example Including a TCA**, page 4-8).

Step 22  Click **OK**. The Soft Properties Manager window is displayed.
Step 23 Click Close. The Prime Network Vision window or Inventory window is displayed, depending on your original selection.

Step 24 To view the newly created soft property:
- Click the top right corner to close the Inventory window, then open it again.
- Move up and down a branch in the tree pane of the Inventory window.

The Inventory window for the required managed element is displayed with the newly defined soft property.

Step 25 In Prime Network Vision:
- Right-click the required managed element in the tree pane or workspace.
- Right-click the required object of the network element in the Inventory window.

Step 26 Choose Management > Soft Properties Management. The Soft Properties Manager window is displayed.

Step 27 Select the required properties panel and soft property in the element properties table.

Step 28 Open the hierarchy manager table in one of the following ways:
- Click Hierarchy Manager in the toolbar.
- Choose Tools > Hierarchy Manager.
- Right-click the item, then choose Hierarchy Manager.

Step 29 Select the node in the hierarchy from which you want to publish the soft property.

Step 30 Click Copy or Cut in the toolbar to copy or cut the soft property.

Step 31 Select the required node in the hierarchy where you want to publish the soft property.

Step 32 Click Paste in the toolbar to paste the soft property. The soft property is published to the selected node in the hierarchy.

Step 33 Click Close. The Soft Properties Manager window is displayed.
For more information about defining a soft property with a TCA, see Soft Property Example Including a TCA, page 4-8.

Soft Property Example Including a TCA

Example 1
To create a soft property including a TCA:

Step 1 Perform Step 1 through Step 20, as described in the Basic Soft Property Example, page 4-1.
Step 2 Select the TCA Alarms tab.
Step 3 Click Add. The General tab of the Add TCA dialog box is displayed.
Step 4 Define the information in the TCA Alarms tab as follows:
   - Name—My value is not 5.
   - Enabled—Selected.
   - Description—Show this alarm if the value is not equal to 5.
   - Alarm Severity—CRITICAL.
Step 5 Select the Trigger tab in the Add TCA dialog box.
Step 6 Define the information in the Trigger tab as follows:
   - Trigger—Value Not Equal.
   - To value—5
Step 7 Click OK. The TCA Alarms tab is displayed with the defined TCA.
Step 8 Click OK. The Soft Properties Manager window is displayed.
Step 9 Click Close. The Prime Network Vision window or the Inventory window is displayed, depending on your original selection.
Step 10 To view the newly created soft property:
   - Click the top right corner to close the Inventory window, then open it again.
   - Move up and down a branch in the tree pane of the Inventory window.
The Inventory window for the required managed element is displayed with the newly defined soft property.
Step 11 Perform Step 24 through Step 33, as described in Basic Soft Property Example, page 4-1.

Example 2
To define a soft property that includes a TCA:

Step 1 Define a new soft property with the telnet command show clock.
Make sure you assign a relatively short polling rate.
Step 2 In the parsing rules section add a parse Integer rule.
Let the start index value be 5, and the length value, 2.
Step 3 In the TCA Alarm tab, create a new alarm definition.
Step 4 Set the trigger to upper threshold with a value of 0 (or any other number representing time in minutes).
Step 5 Click OK. The TCA Alarms tab is displayed with the defined TCA.
Step 6 Click OK. The Soft Properties Manager window is displayed.
Step 7 Click Close. The Prime Network Vision window or the Inventory window is displayed, depending on your selection.
Step 8 To view the newly created soft property:
   • Click the top right corner to close the Inventory window, then open it again.
   • Move up and down a branch in the tree pane of the Inventory window.
Step 9 The Inventory window for the required managed element is displayed with the newly defined soft property.
The alarm will be triggered the next time the Soft Property is sampled and is above the threshold value you assigned.

For information about defining a soft property table, see Soft Property Table Example, page 4-9.

Soft Property Table Example

To create a soft property table:

Step 1 Perform Step 1 and Step 2, as described in the Soft Property Example Including a TCA, page 4-8.
Step 2 Open the Add Soft Property dialog box in one of the following ways:
   • Click New Element in the toolbar.
   • Choose File > New Element.
Step 3 Define the soft property information in the General tab as follows:
   • Name—sp02
   • Label—My Soft Table
   • Description—Example of a soft table
   • Type—Table
   • Polling Rate—Status
   • Enabled—Selected
Step 4 Select the Parsing tab. The Parsing tab is displayed.
Step 5 Define the information in the Parsing tab by entering .1.3.6.1.2.1.4.2 0.1 in the text box.
Step 6 Click Add. The Add Edit Column Controller dialog box is displayed.
Step 7 Define the information in the Add Edit Column Controller dialog box as follows:
   • Column Title—My First Column
   • Column Data—2
Step 8 Click OK. The Parsing tab of the Add Soft Property dialog box is displayed with the defined table information for column 1 of the table.

Step 9 Click Add. The Add Edit Column Controller dialog box is displayed.

Step 10 Define the information in the Add Edit Column Controller dialog box as follows:
- Column Title—My Second Column
- Column Data—3

Step 11 Click OK. The Parsing tab of the Add Soft Property dialog box is displayed with the defined table information for columns 1 and 2 of the table.

**Figure 4-8 Add Soft Property Dialog Box**

Step 12 Click Debug. The Debug Soft Property dialog box is displayed.

Step 13 Click Close. The Add Soft Property dialog box is displayed.

Step 14 Click OK. The Soft Properties Manager window is displayed.

Step 15 Click Close. The Prime Network Vision window or the Inventory window is displayed, depending on your original selection.

Step 16 To view the newly created soft property:
- Click the top right corner to close the Inventory window, then open it again.
- Move up and down a branch in the tree pane of the Inventory window.

The Inventory window for the required managed element is displayed with the newly defined soft property table.

Step 17 Perform Step 24 through Step 33, as described in the Basic Soft Property Example, page 4-1.
This chapter describes 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. The chapter includes the name, description, expected input, validation rules, and any unique fields displayed in the dialog box for each operator and provides an example, as well.

Topics include:

- **Header and Footer, page 5-1**
- **Remove Lines, page 5-3**
- **Select Lines, page 5-4**
- **Replace, page 5-5**
- **Match, page 5-6**
- **Set, page 5-7**
- **Substring, page 5-8**
- **Parse Integer, page 5-9**

For more information about the Add/Edit Parsing Rule dialog box, see Defining the Soft Property Parsing Parameters, page 3-9.

### Header and Footer

**Description**

Removes a specified number of lines from the header and footer of the input text.

**Dialog Box**

*Figure 5-1* shows the Add/Edit Parsing Rule dialog box that is displayed when the Header And Footer operator is selected. In addition, the dialog box displays an example using the Header And Footer operator.
Figure 5-1   Header and Footer Operator

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Validation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header lines</td>
<td>The number of header lines to be removed.</td>
<td>Integers only. Mandatory.</td>
</tr>
<tr>
<td>Footer lines</td>
<td>The number of footer lines to be removed.</td>
<td>Integers only. Mandatory.</td>
</tr>
</tbody>
</table>
Remove Lines

Description

Removes a range of lines from the specified starting row to the specified end row of the input text.

Dialog Box

Figure 5-2 shows the Add/Edit Parsing Rule dialog box that is displayed when the Remove Lines operator is selected. In addition, the dialog box displays an example using the Remove Lines operator.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Validation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>From line</td>
<td>Index of first row to begin removal, inclusive.</td>
<td>Integer only. Mandatory.</td>
</tr>
<tr>
<td>To line</td>
<td>Index of last row to be removed, inclusive.</td>
<td>Integer only. Equal to or greater than From line. Mandatory.</td>
</tr>
</tbody>
</table>
Select Lines

Description

Extracts a range of lines from the specified starting row to the specified end row of the input text.

Dialog Box

Figure 5-3 shows the Add/Edit Parsing Rule dialog box that is displayed when the Select Lines operator is selected. In addition, the dialog box displays an example using the Select Lines operator.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Validation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>From line</td>
<td>Index of first row to begin selection, inclusive.</td>
<td>Integer only. Mandatory.</td>
</tr>
<tr>
<td>To line</td>
<td>Index of last row to be selected, inclusive.</td>
<td>Integer only. Equal to or greater than From line. Mandatory.</td>
</tr>
</tbody>
</table>
Replace

Description

Finds one substring or all occurrences of a substring that matches a specified regular expression, and replaces it with a specified value.

Dialog Box

Figure 5-4 shows the Add/Edit Parsing Rule dialog box that is displayed when the Replace operator is selected. In addition, the dialog box displays an example using the Replace operator.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Validation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Search for value or regular expression.</td>
<td>Text. Mandatory.</td>
</tr>
<tr>
<td>With</td>
<td>Replace string with value or regular expression.</td>
<td>Text. Mandatory.</td>
</tr>
<tr>
<td>From Index</td>
<td>Starting index.</td>
<td>Integer. Mandatory.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> The value entered in this field must be 1 or higher.</td>
<td></td>
</tr>
<tr>
<td>Replace All</td>
<td>Check box. Select this option to replace all occurrences of the matching substrings, otherwise only the first instance is replaced.</td>
<td>Default is unchecked.</td>
</tr>
</tbody>
</table>
Match

Description

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 what you choose in the If no match drop down list.

Dialog Box

Figure 5-5 shows the Add/Edit Parsing Rule dialog box that is displayed when the Match operator is selected. In addition, the dialog box displays an example using the Match operator.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Validation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Search for value or regular expression.</td>
<td>Text. Mandatory.</td>
</tr>
<tr>
<td>If no match</td>
<td>Returns the original input or and empty string, based on your choice.</td>
<td></td>
</tr>
</tbody>
</table>
Set

Description

Prints the results of the input and output buffers.

Dialog Box

Figure 5-6 shows the Add/Edit Parsing Rule dialog box that is displayed when the Set operator is selected. In addition, the dialog box displays an example using the Set operator.

Figure 5-6  Set Operator

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Validation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>Regular expression template to use for formatting. $_$ specifies the main output buffer.</td>
<td>Text. Mandatory.</td>
</tr>
</tbody>
</table>
Substring

Description

Extracts a substring of a specified length from a specified starting point.

Dialog Box

Figure 5-7 shows the Add/Edit Parsing Rule dialog box that is displayed when the Substring operator is selected. In addition, the dialog box displays an example using the Substring operator.

![Substring Operator](image)

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Validation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Index</td>
<td>Begin index to select.</td>
<td>Integer. Mandatory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> The value entered in this field must be 1 or higher.</td>
</tr>
<tr>
<td>Length</td>
<td>How many characters to select.</td>
<td>Integer. Mandatory.</td>
</tr>
</tbody>
</table>
## Parse Integer

### Description

Uses the substring rule, and 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.

### Dialog Box

Figure 5-8 shows the Add/Edit Parsing Rule dialog box that is displayed when the Parse Integer operator is selected. In addition, the dialog box displays an example using the Parse Integer operator.

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Validation Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>From Index</td>
<td>Starting index to select.</td>
<td>Integer. Mandatory.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> The value entered in this field must be 1 or higher.</td>
</tr>
<tr>
<td>To Index</td>
<td>Ending index to select.</td>
<td>Integer. Mandatory.</td>
</tr>
</tbody>
</table>
Parse Integer
Alarm Threshold Triggers

This chapter describes the predefined alarm threshold triggers available for defining TCAs that are displayed in the Trigger tab of the Add TCA dialog box. This chapter includes the name, description, and unique fields displayed in the dialog box for each alarm threshold trigger.

You can define multiple alarms for the same soft property. The alarm is displayed in the ticket pane of the Prime Network Vision window.

A counter value, as described in this chapter, is a numeric value that always increases.

Topics include:

- Value Equal, page 6-1
- Value Not Equal, page 6-2
- Upper Threshold, page 6-2
- Lower Threshold, page 6-2
- Upper Rate, page 6-3
- Lower Rate, page 6-3

For more information about the Add TCA dialog box, see Defining TCA Parameters, page 3-14.

Value Equal

The alarm condition is reached when the soft property value is equal to the value defined in Alarm Value regardless of whether or not it is numeric.

When the Value Equal threshold type is selected from the Trigger drop-down list of the Add TCA dialog box, the Trigger tab is displayed.

The following fields are displayed in the Trigger tab when the Value Equal threshold type is selected:

- To value—The target value.

- Trigger alarm only if change persists more than—Select this option to trigger the alarm if the alarm criteria persist for the defined period. The time period is defined in seconds in the trigger alarm field. For example, if CPU usage is over 85% (the alarm criteria) and this condition persists for more than one minute (the defined period), then the alarm is triggered. When setting the values for this option, bear in mind that if the soft property polls the device every x seconds, entering a defined period for less than x provides no benefit.
Value Not Equal

The alarm condition is reached when the soft property value is not equal to the value defined in Alarm Value, regardless of whether or not it is numeric.

When the Value Not Equal threshold type is selected from the Trigger drop-down list of the Add TCA dialog box, the Trigger tab is displayed.

The following fields are displayed in the Trigger tab when the Value Not Equal threshold type is selected:

- To value—The target value.
- Trigger alarm only if change persists more than—Select this option to trigger the alarm if the alarm criteria persist for the defined period. The time period is defined in seconds in the trigger alarm field.

Upper Threshold

When it is crossed, the upper threshold value triggers the alarm for the defined numeric properties.

This threshold trigger must receive a numeric value. To receive a numeric value, the parse integer rule must be applied on the soft property as an ending rule. For more information about parsing integers, see Chapter 5, “Parsing Operators/Rules.”

When the Upper Threshold type is selected from the Trigger drop-down list of the Add TCA dialog box, the Trigger tab is displayed.

The following fields are displayed in the Trigger tab when the Upper Threshold type is selected:

- Trigger alarm when value is above—The value which, when crossed, generates the alarm.
- Clear alarm when value is below—The value which, when crossed (when going back), clears the alarm.
- Trigger alarm only if change persists more than—Select this option to trigger the alarm if the alarm criteria persist for the defined period. The time period is defined in seconds in the trigger alarm field.

Lower Threshold

When it is crossed, the lower threshold value triggers the alarm for the defined numeric properties.

This threshold trigger must receive a numeric value. To receive a numeric value, the parse integer rule must be applied on the soft property as an ending rule. For more information about parsing integers, see Chapter 5, “Parsing Operators/Rules.”

When the Lower Threshold type is selected from the Trigger drop-down list of the Add TCA dialog box, the Trigger tab is displayed.

The following fields are displayed in the Trigger tab when the Lower Threshold type is selected:

- Trigger alarm when value is below—The value which, when crossed, generates the alarm.
Upper Rate

The upper rate trigger is used to check the counter value changes over a period of one second. When the specified rate is crossed, it triggers the alarm for the defined numeric property. When this is used with the Trigger alarm only if change persists more than option, described in Upper Threshold, page 6-2, you can check that the rate is maintained above the specified value over time.

Note

The following is the calculation for the rate every one second: If the property is sampled every $x$ seconds, the calculation is the \textit{current value} less the \textit{previous value} divided by $x$ seconds.

This threshold trigger must receive a numeric value. To receive a numeric value, the parse integer rule must be applied on the soft property as an ending rule. For more information about parsing integers, see Chapter 5, “Parsing Operators/Rules.”

When the Upper Rate threshold type is selected from the Trigger drop-down list of the Add TCA dialog box, the Trigger tab is displayed.

The following fields are displayed in the Trigger tab when the Upper Rate threshold type is selected:

- Trigger alarm when value is above—The value which, when crossed, generates the alarm.
- Clear alarm when value is below—The value which, when crossed (when going back), clears the alarm.
- Trigger alarm only if change persists more than—Select this option to trigger the alarm if the alarm criteria persist for the defined period. This period is defined in seconds in the trigger alarm field.

Lower Rate

The lower rate trigger is used to check the counter value changes over a period of one second. When the specified rate is crossed, it triggers the alarm for the defined numeric property. When this is used with the Trigger alarm only if change persists more than option, described in Upper Threshold, page 6-2, you can check that the rate is maintained below the specified value over time.

Note

The following is the calculation for the rate every one second: If the property is sampled every $x$ seconds, the calculation is the \textit{current value} less the \textit{previous value} divided by $x$ seconds.

This threshold trigger must receive a numeric value. To receive a numeric value, the parse integer rule must be applied on the soft property as an ending rule. For more information about parsing integers, see Chapter 5, “Parsing Operators/Rules.”
When the Lower Rate threshold type is selected from the Trigger drop-down list of the Add TCA dialog box, the Trigger tab is displayed.

The following fields are displayed in the Trigger tab when the Lower Rate threshold type is selected:

- **Trigger alarm when value is below**—The value which, when crossed, generates the alarm.
- **Clear alarm when value is above**—The value which, when crossed (when going back), clears the alarm.
- **Trigger alarm only if change persists more than**—Select this option to trigger the alarm if the alarm criteria persist for the defined time period. This period is defined in seconds in the trigger alarm field.
Regular Expressions for Soft Properties Manager

This section is based on the documentation of the package GNU RegExp.

A regular expression consists of a character string in which some characters are given special meaning with regard to pattern matching. Regular expressions have been in use from the early days of computing, and provide a powerful and efficient way to parse, interpret, and search and replace text within an application.

Topics include:

- **Supported Syntax, page 7-1**
- **Unsupported Syntax, page 7-4**

### Supported Syntax

Within a regular expression, the following characters have special meaning.

#### Positional Operators

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>^</td>
<td>Matches at the beginning of a line.</td>
</tr>
<tr>
<td>$</td>
<td>Matches at the end of a line.</td>
</tr>
<tr>
<td>\A</td>
<td>Matches the start of the entire string.</td>
</tr>
<tr>
<td>\Z</td>
<td>Matches the end of the entire string.</td>
</tr>
<tr>
<td>\b</td>
<td>Matches at a word break (Perl5 syntax only).</td>
</tr>
<tr>
<td>\B</td>
<td>Matches at a nonword break (opposite of \b) (Perl5 syntax only).</td>
</tr>
<tr>
<td>&lt;</td>
<td>Matches at the start of a word (egrep syntax only).</td>
</tr>
<tr>
<td>&gt;</td>
<td>Matches at the end of a word (egrep syntax only).</td>
</tr>
</tbody>
</table>
### One-Character Operators

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.</td>
<td>Matches any single character.</td>
</tr>
<tr>
<td>\d</td>
<td>Matches any decimal digit.</td>
</tr>
<tr>
<td>\D</td>
<td>Matches any nondigit.</td>
</tr>
<tr>
<td>\n</td>
<td>Matches a newline character.</td>
</tr>
<tr>
<td>\r</td>
<td>Matches a return character.</td>
</tr>
<tr>
<td>\s</td>
<td>Matches any whitespace character.</td>
</tr>
<tr>
<td>\S</td>
<td>Matches any nonwhitespace character.</td>
</tr>
<tr>
<td>\t</td>
<td>Matches a horizontal tab character.</td>
</tr>
<tr>
<td>\w</td>
<td>Matches any word (alphanumeric) character.</td>
</tr>
<tr>
<td>\W</td>
<td>Matches any nonword (alphanumeric) character.</td>
</tr>
<tr>
<td>\x</td>
<td>Matches the character (x), if (x) is not one of the above listed escape sequences.</td>
</tr>
</tbody>
</table>

### Character Class Operator

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[abc]</td>
<td>Matches any character in the set (a, b,) or (c).</td>
</tr>
<tr>
<td>[^abc]</td>
<td>Matches any character not in the set (a, b,) or (c).</td>
</tr>
<tr>
<td>[a-z]</td>
<td>Matches any character in the range (a) to (z), inclusive.</td>
</tr>
</tbody>
</table>

Leading or trailing dash interpreted literally.

Within a character class expression, the following sequences have special meaning if the syntax bit RE_CHAR_CLASSES is enabled:

<table>
<thead>
<tr>
<th>Character Sequences</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:alnum:]</td>
<td>Any alphanumeric character.</td>
</tr>
<tr>
<td>[:alpha:]</td>
<td>Any alphabetic character.</td>
</tr>
<tr>
<td>[:blank:]</td>
<td>A space or horizontal tab.</td>
</tr>
<tr>
<td>[:cntrl:]</td>
<td>A control character.</td>
</tr>
<tr>
<td>[:digit:]</td>
<td>A decimal digit.</td>
</tr>
<tr>
<td>[:graph:]</td>
<td>A nonspace, noncontrol character.</td>
</tr>
<tr>
<td>[:lower:]</td>
<td>A lowercase letter.</td>
</tr>
<tr>
<td>[:print:]</td>
<td>Same as graph, but also space and tab.</td>
</tr>
<tr>
<td>[:punct:]</td>
<td>A punctuation character.</td>
</tr>
<tr>
<td>[:space:]</td>
<td>Any whitespace character, including newline and return.</td>
</tr>
</tbody>
</table>
Chapter 7  Regular Expressions for Soft Properties Manager

Supported Syntax

Subexpressions and Back References

<table>
<thead>
<tr>
<th>Character Sequences</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[:upper:]</td>
<td>An uppercase letter.</td>
</tr>
<tr>
<td>[:xdigit:]</td>
<td>A valid hexadecimal digit.</td>
</tr>
</tbody>
</table>

Branching (Alternation) Operator

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(abc)</td>
<td>Matches whatever the expression abc would match, and saves it as a subexpression. Also used for grouping.</td>
</tr>
<tr>
<td>(?:...)</td>
<td>Pure grouping operator; does not save contents.</td>
</tr>
<tr>
<td>(?#...)</td>
<td>Embedded comment; ignored by engine.</td>
</tr>
<tr>
<td>\n</td>
<td>Where 0 &lt; n &lt; 10, matches the same thing the nth subexpression matched.</td>
</tr>
</tbody>
</table>

Repeating Operators

These symbols operate on the previous atomic expression:

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>Matches the preceding expression or the null string.</td>
</tr>
<tr>
<td>*</td>
<td>Matches the null string or any number of repetitions of the preceding expression.</td>
</tr>
<tr>
<td>+</td>
<td>Matches one or more repetitions of the preceding expression.</td>
</tr>
<tr>
<td>{m}</td>
<td>Matches exactly m repetitions of the one-character expression.</td>
</tr>
<tr>
<td>{m,n}</td>
<td>Matches between m and n repetitions of the preceding expression, inclusive.</td>
</tr>
<tr>
<td>{m,}</td>
<td>Matches m or more repetitions of the preceding expression.</td>
</tr>
</tbody>
</table>

Stingy (Minimal) Matching

If a repeating operator (above) is immediately followed by a question mark (?), the repeating operator stops at the smallest number of repetitions that can complete the rest of the match.
Lookahead

Lookahead refers to the ability to match part of an expression without consuming any of the input text. There are two variations to this:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(?=foo)</td>
<td>Matches at any position where foo would match, but does not consume any characters of the input.</td>
</tr>
<tr>
<td>(?!foo)</td>
<td>Matches at any position where foo would not match, but does not consume any characters of the input.</td>
</tr>
</tbody>
</table>

Unsupported Syntax

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:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(?mods)</td>
<td>Inlined compilation/execution modifiers (Perl5).</td>
</tr>
<tr>
<td>\G</td>
<td>End of previous match (Perl5).</td>
</tr>
<tr>
<td>[symbol.]</td>
<td>Collating symbol in class expression (POSIX).</td>
</tr>
<tr>
<td>[=class=]</td>
<td>Equivalence class in class expression (POSIX).</td>
</tr>
<tr>
<td>s/foo/bar</td>
<td>Style expressions as in sed and awk.</td>
</tr>
</tbody>
</table>
PART 3

Prime Network Workflow

This part contains the following chapters:

- Chapter 8, “Getting Started with Prime Network Workflow”
- Chapter 9, “Working with Prime Network Workflow”
- Chapter 10, “Managing Workflows”
- Chapter 11, “Customizing Prime Network Workflow”
Getting Started with Prime Network Workflow

This chapter introduces Prime Network Workflow and its concepts and describes the taskflow for creating, testing, deploying, running, and viewing a workflow.

Topics include:
- About Prime Network Workflow, page 8-1
- Workflow Concepts, page 8-2
- Taskflow for Creating a Workflow, page 8-4

Additional information about Prime Network Workflow is available on the Cisco Developer Network (CDN)

About Prime Network Workflow

Prime Network Workflow, which is based on LiquidBPM by Autonomy, Inc., is used to create and run the logical flows of activation commands, including complex rollback scenarios. This logic enables you to define relationships between tasks, including sequences, branches, failure procedures, and access to Prime Network commands as well as the information model. Prime Network Workflow can interface with an external system such as an order management system to create a full service-provisioning solution that is user-customizable and user-extendable.

Prime Network Workflow is a process management GUI that acts as a powerful visual design and execution tracing tool for defining and deploying activation workflows. A workflow consists of several tasks grouped and arranged in a hierarchy. Workflow management is supported in runtime, and includes a runtime GUI control console.

Prime Network Workflow is used to construct workflows which run gateway commands and provides complete access to the Prime Network live network information model. Prime Network Workflow provides a nested structure. Workflow commands are also available through the Prime Network API.

The workflow engine resides on the gateway using AVM 66. All deployed workflows are stored on the gateway. After a workflow is deployed, it is accessible through Prime Network Administration for viewing properties and status. Deployed workflows can be invoked through the Prime Network API using BQL. The workflow engine provides default workflow inherent rollback.

In addition, you can view a history of the invoked workflows using Prime Network Events.
Workflow Concepts

This section describes the following workflow concepts:

- Workflow Task, page 8-2
- BQL Task, page 8-2
- Lock/Unlock Task, page 8-3
- Workflow Call (Synchronous Workflow Nesting), page 8-3

Workflow Task

Tasks are added to workflows to define the processes. Each task performs a specific function. Tasks can be quickly added and configured using Prime Network Workflow.

Tasks can be classed as predecessor or successor tasks:

- A predecessor task is one that must be completed before the next task can be executed.
- A successor task is one that is run after a predecessor task.

For example, if a workflow consists of two tasks, Configure Switch (first task) and Configure Router (second task), the Configure Switch task is the predecessor task and must be completed before the Configure Router (successor task) can be run.

An activation state is associated with each task:

- Ready—The task is ready to begin when the constraints (for example, start time or predecessors) have been satisfied.
- Active—The task is being run.
- Done—The task is complete.
- Abort—The task has failed or the state has been set manually. The task can be manually reset to Ready or Done.
- Passive—The task exists but is no longer relevant. For the purposes of successive tasks, the passive task is considered done.

Figure 8-1 presents the typical task sequence.

BQL Task

The Execute BQL task is found in Prime Network Workflow toolbar. This task runs the BQL command specified in the Command Template tab of the Task Properties dialog box, and stores the result in a task attribute called Result so that it can be used by scripts and other tasks.
The Command Template tab can reference workflow attributes and task attributes. At runtime, the attribute’s values are substituted into the template before it is run.

**Note**
The BQL task in a workflow can only execute script BQLs and any of the generic commands: Get, Delete, Find, Update, Set.

**Note**
When you execute a workflow to run commands on a device, using the RunWorkflow BQL command, it will be executed only if the preview flag in the RunWorkflow BQL is *false*. If the value of the preview flag is *true*, it will only be previewed. See Running a Workflow, page 9-22 for details.

## Lock/Unlock Task

The Lock task has two main objectives:

- To allow workflow instances to declare the resources that they use and the scope of their usage.
- To ensure that those resources are not used by any other workflow instance during that scope.

The Lock task enables you to protect any component from concurrent use by multiple workflows. You can lock an object that represents a single resource and guard access to it. A resource’s identifier serves as the name of the lock. At any given time, a lock can be owned by, at most, only one workflow. Resources can be automatically locked during rollback.

The system prevents deadlocks before they occur. Upon detecting an imminent deadlock, the lock operation belonging to the workflow with the least progress fails. A failed lock might or might not abort the workflow.

The locking mechanism does not cover every access to every resource. Only workflows participate in the locking process. Nonworkflow activities can access a resource even when it is locked by a workflow. Participation in the locking process is optional.

You can:

- Lock or unlock single or multiple resources.
- Unlock resources when a workflow terminates.
- Lock resources during rollback.

## Workflow Call (Synchronous Workflow Nesting)

Synchronous workflow nesting allows workflow designers to invoke sub-workflows synchronously and pass arguments to each subworkflow invocation. This enables the workflow designer to isolate the tasks running in the subworkflow as much as possible from the tasks running in the parent workflow and in other subworkflows.

The following functionality is available:

- The child workflow is run in a separate workflow. The parent workflow waits for the child workflow to terminate.
- When a child workflow is stopped, it causes its parent workflow to stop as well.
- The child workflow has a separate scope for attributes.
- The output of the child workflow is directed to the parent workflow.
- The parent workflow can pass parameters to its child workflow.
- The correct rollback sequence is maintained throughout the depth of the lineage.
- Child workflows are not visible through the API. You interact directly with the parent.
- Delete and abort operations on parent workflows are delegated to child workflows.

**Note**
The maximum workflow nesting depth is defined in the registry. The default value is 16.

### Taskflow for Creating a Workflow

Figure 8-2 presents the process required when working with and managing workflows. The tasks are described in the order in which they must be performed.
1. Create a command using Command Builder and preview it. For more information, see Creating a Command, page 13-7.

2. Define tasks and workflows using Prime Network Workflow. For more information, see Chapter 9, “Working with Prime Network Workflow.”

3. Set up the command descriptor scripts and add workflow attributes:
   a. Copy the command descriptor scripts—Copy the command descriptor scripts that were defined using Command Builder to Prime Network Workflow BQL Tasks command template tabs.
   b. Edit the command descriptor scripts—Edit the command descriptor scripts that were defined using Command Builder to Prime Network Workflow BQL Tasks command template tabs, as required.
   c. Add workflow and task attributes—Add workflow and task attributes using Prime Network Workflow.

4. Test the workflow locally. For more information, see Testing a Workflow, page 9-20.

5. Deploy the workflow on the gateway. For more information, see Deploying a Workflow Template, page 9-5.
6. Run the workflow using a BQL command. For more information, see Running a Workflow, page 9-22.

7. Manage and view the workflow in Prime Network Administration. For more information, see Chapter 10, “Managing Workflows.”

8. View workflow results in Prime Network Events. Prime Network Events enables you to view the workflow history, including when workflows have been completed, their status, and the command invoked.

Note  For Prime Network Events to display an event when a Workflow is aborted, you must add the following line to the preActivateScript in the Callback Scripts tab:

```javascript
thisWorkflow.getRootTask().abort();
```

For more information, see Prime Network Workflow application online help for Engine Behavior > Workflow State Transitions.
CHAPTER 9

Working with Prime Network Workflow

This chapter provides instructions for launching the Prime Network Workflow application, describes the Prime Network Workflow working environment, and explains how to operate the customized functionality.

Topics include:
- Starting Prime Network Workflow, page 9-1
- Prime Network Workflow Window, page 9-3
- Creating a New Workflow Template, page 9-4
- Retrieving a Workflow Template, page 9-5
- Deploying a Workflow Template, page 9-5
- Running a Workflow Template, page 9-6
- Deleting a Workflow Template, page 9-6
- Viewing Workflow Properties, page 9-6
- Working with the Task Library, page 9-8
- Testing a Workflow, page 9-20
- Gateway Workflow Commands and Operations, page 9-22

Starting Prime Network Workflow

This section provides instructions for launching Prime Network Workflow. Prime Network Workflow is password-protected to ensure security. Before using Prime Network Workflow, make sure you know your username, password, and the gateway IP address or hostname. Installation of Prime Network Workflow is optional. Prime Network Workflow can be installed using the regular client installation package. For more information, see the Cisco Prime Network 3.8 Installation Guide.

When the client opens, it negotiates with the server to validate the account credentials. If the credentials are not validated, Prime Network Workflow does not open.

Note

Deploy, retrieve, and delete operations work only if AVM 66 is up and running. However, executing a workflow using Prime Network Workflow is independent of AVM 66, and the workflow gets activated even if AVM 66 is down.
To start Prime Network Workflow:

**Step 1** From the Start menu, choose **Programs > Cisco Prime Network > Cisco Prime Network Workflow**. The Prime Network Workflow Login dialog box is displayed.

**Step 2** Enter your username and password.

**Step 3** Specify the gateway in one of the following ways:
- Enter the gateway information in the Host field as an IP address or hostname.
- Choose a gateway from the Host drop-down list.

**Note** The gateway IP address or hostname that was used when you last logged in automatically appears at the top of the host list.

**Step 4** Click **OK**. The Prime Network Workflow window is displayed (see Figure 9-1).

**Note** The Prime Network Workflow window appears empty when it is opened. You can create a new workflow or retrieve a workflow that was created previously.
Prime Network Workflow Window

Figure 9-1 shows the Prime Network Workflow window with a template.

![Workflow Window Diagram]

The Prime Network Workflow window displays information on the workflow and the tasks within the workflow.

Table 9-1 identifies the icons and actions that are available in the Prime Network Workflow toolbar and the Tools menu.

### Table 9-1 Toolbar and Tool Menu Options

<table>
<thead>
<tr>
<th>Button</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Retrieves" /></td>
<td>Retrieves (loads) a workflow from the gateway.</td>
</tr>
<tr>
<td><img src="image2.png" alt="Deletes" /></td>
<td>Deletes a workflow from the gateway.</td>
</tr>
<tr>
<td><img src="image3.png" alt="Deploys" /></td>
<td>Deploys (uploads) the workflow on the gateway.</td>
</tr>
</tbody>
</table>
Creating a New Workflow Template

The workflow template creation process begins with creating a new workflow template with a unique name. A blank template is created and opened for editing. You then add tasks to the template to create the logical flow that is required. The workflow template is stored locally, and each update is automatically saved.

Note
You should save templates to a disk using the Export button only.

Workflow Template Names

When you create the unique name for each workflow template, do not include the following wildcard characters:

- Underscore (_): to denote a single character.
- Percent (%): to denote a zero or many characters.

If you include _ and % in the workflow template, the following message appears in the AVM66 log when you try to run the template or reference it in a subflow:

"WARN [13 21:00:08,248] - dralasoft.workflow - Task aborted. Task: 245886, Workflow: 245885 java.lang.IllegalArgumentException: Template AA_BB.template is ambiguous, templates ids are: 245874, 245873"

The following examples show workflow template names that can lead to ambiguity if they are deployed together:

- WFTLM_MUESTRA.template and WFTLM#MUESTRA.template
- WFTLM%MUESTRA.template and WFTLM####MUESTRA.template

The ambiguity only occurs when the template containing the wildcard characters is run.

---

Table 9-2 identifies additional icons and actions that are available in the toolbar.

### Table 9-2 Additional Toolbar Options

<table>
<thead>
<tr>
<th>Icon</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execute BQL</td>
<td>Execute BQL is used to invoke commands previously created and residing on the server. This is the main task used for implementing an activation workflow.</td>
</tr>
<tr>
<td>Lock</td>
<td>Lock is used to obtain a lock on a resource.</td>
</tr>
<tr>
<td>Unlock</td>
<td>Unlock is used to unlock a resource.</td>
</tr>
<tr>
<td>Workflow Call</td>
<td>Workflow Call is used to synchronically execute or call another workflow.</td>
</tr>
<tr>
<td>Subflow</td>
<td>Subflow is used to embed and execute another workflow.</td>
</tr>
</tbody>
</table>
Workflow Attribute Types

By default, all attributes in Prime Network Workflow are of the type String. To use another attribute type, such as Integer, set the flag use-workflow-string-param-casting to true in /export/home/ana37/Main/registry/workflowavm.xml. In workflowavm.xml, the flag use-workflow-string-param-casting appears under the workflow key.

After you change this flag to true, you can use attribute types other than String.

Note
Changes to the registry should be performed only with the support of Cisco. For details, contact your Cisco account representative.

Retrieving a Workflow Template

You can retrieve a workflow template that was previously created and deployed on the gateway or server. Once the workflow template has been retrieved, you can do one of the following:

- Edit the workflow template.
- Deploy the workflow template.
- Execute the workflow template.
- Delete the workflow template.

To retrieve a workflow template:

**Step 1**
On the toolbar, click **Retrieve Workflow**.

The Retrieve Workflow Template from Server dialog box displays the list of existing workflow templates.

**Step 2**
Choose the workflow template that you want to load.

**Step 3**
Click **OK**. A confirmation message is displayed.

**Step 4**
Click **OK**. The required workflow template is opened and displayed in the Prime Network Workflow window.

Deploying a Workflow Template

After you have tested the workflow template locally and are satisfied with it, you can deploy it to the Prime Network server (remote server), where it becomes available to all authorized users.

To deploy a workflow template:

**Step 1**
In the toolbar, click **Deploy Workflow**. A confirmation message is displayed.

**Step 2**
Click **Yes**. A success message is displayed.
Running a Workflow Template

Running a workflow template requires executing a BQL command. See Running a Workflow, page 9-22 for details.

Deleting a Workflow Template

You can delete a workflow template from the server.

A workflow template does not have to be open or displayed in the Prime Network Workflow window before it can be deleted.

To delete a workflow template:

1. In the toolbar, click **Delete Workflow from Server**. The Delete Workflow Template from Server dialog box is displayed.
2. Choose the template that you want to delete.
3. Click **OK**. A success message is displayed.
4. Click **OK**. The workflow template is deleted from the local server.

Viewing Workflow Properties

The Workflow Properties dialog box enables you to view the workflow callback scripts. You can view the properties of a workflow.

To view workflow properties:

1. Create or retrieve the required workflow template.
2. Right-click the drawing area to display the menu, then choose **Workflow Properties**. The Workflow Properties dialog box for the required workflow template is displayed (Figure 9-2).
Step 3

Choose the Attributes tab. The attributes of the workflow are displayed (Figure 9-3).

Figure 9-3    Attributes Tab

![Attributes Tab Image]
Step 4  Choose the Callback Scripts tab (Figure 9-4).

![Callback Scripts Tab](image)

The Select Script list enables you to choose one of the following options to activate the script:

- **preActiveScript**—Activates the script before the workflow is active.
- **activeScript**—Activates the script when the workflow becomes active.
- **doneScript**—Activates the script when the workflow is successfully completed.
- **exceptionScript**—Activates the script if one of the tasks in the workflow fails.

The Execute button runs the script for testing purposes.

Step 5  Click OK. The Workflow Properties dialog box is closed.

---

### Working with the Task Library

This section describes viewing the properties of the following tasks:

- **Execute BQL Task**, page 9-9
- **Workflow Call Task**, page 9-11
- **Subflow Task**, page 9-14
- **Lock Task**, page 9-16
- **Unlock Task**, page 9-18
Execute BQL Task

The BQL task in a workflow can only execute script BQLs and any of the generic commands: Get, Delete, Find, Update, Set.

To view the properties of an Execute BQL task:

Step 1 Create or retrieve the required workflow template.
Step 2 Choose the required BQL task in the drawing area.
Step 3 Right-click the BQL task, then choose Task Properties. The Task Properties dialog box for the required task is displayed (Figure 9-5).

Figure 9-5  Task Properties Dialog Box - General Properties Tab

Note The Earliest Start option is not supported.
Step 4  
Choose the Attributes tab. The attributes of the task are displayed (Figure 9-6).

![Attributes Tab](image)

**Figure 9-6  Attributes Tab**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Default Value</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>VrfName</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>mask</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>port</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>vci</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>vpi</td>
<td>String</td>
<td></td>
</tr>
</tbody>
</table>

Use the Command Template tab to enter the command template XML to create a command instance descriptor (CID) for running the gateway command or activation script.

Step 5  
Choose the Command Template tab (Figure 9-7).

![Command Template Tab](image)

**Figure 9-7  Command Template Tab**
The format of the template is similar to a CID, with the option for setting parameters that would be evaluated on execution:

- $\text{Attribute name}$—Is evaluated by that name from a workflow attribute.
- $\text{Task name:Attribute name}$—Is evaluated by that name by a task attribute.

If an invalid attribute or task name is entered, the evaluation of the command template fails and stops the task. The following error message is displayed:

```
Failed to create command, task [Task] not performed, aborting task.
Reason: Attribute <Attribute name> evaluation failed, aborting command creation
```

**Note**

Each BQL task includes a task attribute called RollbackEnabled. This attribute defines whether or not this BQL task command is rolled back if the workflow stops. The default value is True.

**Step 6**

Click **OK**. The Task Properties dialog box closes.

---

**Workflow Call Task**

The Workflow Call task is used to synchronically run or call another workflow, which is recognized as a child of this workflow.

**Note**

The maximum workflow nesting depth is defined in the registry. The default value is 16.

To view the properties of a Workflow Call task:

**Step 1**

Create or retrieve the required workflow template.

**Step 2**

Choose the required Workflow Call task in the drawing area.
### Step 3

Right-click the task, then choose **Task Properties**. The Task Properties dialog box for the required task is displayed (Figure 9-8).

**Figure 9-8 Task Properties Dialog Box - Assign Attributes Tab**

The Assign Attributes tab enables you to pass parameters as attributes to child workflows. An attribute can be assigned to a child workflow in two ways:

- By specifying a workflow attribute in a parent workflow, as follows:
  ```plaintext```
  attribute_in_child_workflow=attribute_in_parent_workflow
  ```plaintext```

- By specifying a task in a parent workflow followed by a colon (:) and the attribute name, as follows:
  ```plaintext```
  attribute_in_child_workflow=task_in_parent_workflow:attribute_name
  ```plaintext```
**Chapter 9      Working with Prime Network Workflow**

**Working with the Task Library**

**Step 4**  Choose the General Properties tab. The general properties of the task are displayed (Figure 9-9).

*Figure 9-9    General Properties Tab*

![General Properties Tab](image)

*Note*  The Earliest Start option is not supported.
Step 5  Choose the Attributes tab. The attributes of the task are displayed.

Step 6  Choose the Target Template tab (Figure 9-10).

**Figure 9-10  Target Template Tab**

![Target Template Tab](image)

The Target Template tab enables you to choose the workflow template that is to be defined as the child workflow.

Step 7  Click OK. The Task Properties dialog box is closed.

---

**Subflow Task**

The Subflow task is used to embed and run another workflow in a workflow instance.

**Note**

The maximum workflow nesting depth is defined in the registry. The default value is 16.

To view the properties of a Subflow task:

Step 1  Create or retrieve the required workflow template.

Step 2  Choose the required Subflow task in the Drawing Area.
Step 3  Right-click the task, then choose Task Properties. The Task Properties dialog box for the required task is displayed (Figure 9-11).

![General Properties Tab - Subflow](image)

**Note**  The Earliest Start option is not supported.

Step 4  Choose the Attributes tab. The attributes of the task are displayed.
Step 5  Choose the Target Template tab (Figure 9-12).

![Figure 9-12 Target Template Tab - Subflow](image)

The Target Template tab enables you to choose the workflow template that is to be embedded.

Step 6  Click OK. The Task Properties dialog box is closed.

---

**Lock Task**

The Lock task is used to create a lock on specific resources.

To view the properties of a Lock task:

Step 1  Create or retrieve the required workflow template.

Step 2  Choose the required lock task in the drawing area.
Step 3  Right-click the task, then choose Task Properties. The Task Properties dialog box for the required task is displayed (Figure 9-13).

Figure 9-13   General Properties Tab - Lock

![General Properties Tab](image)

Note  The Earliest Start option is not supported.

Step 4  Choose the Attributes tab. The attributes of the task are displayed.
Step 5  Choose the Resource Names tab (Figure 9-14).

Figure 9-14  Resource Names Tab - Lock

The Resource Names tab enables you to specify the resources that should be locked.

Note  It is also possible to lock multiple resources.

In the text area, enter the names of the resources to be locked. Each resource name should be on a separate line.

Resource names can be passed as parameters that are resolved during task execution to either workflow or task attributes:

- $Attribute name$—Is evaluated from a workflow attribute by that name.
- $Task name:Attribute name$—Is evaluated by a task attribute by that name.

The result of the lock task is stored in an attribute named Result as one of the following:

- Success - lock successfully obtained.
- Failed due to timeout - lock failed due to timeout.
- Failed due to deadlock - lock failed due to a deadlock.

Step 6  Click OK. The Task Properties dialog box is closed.

Unlock Task

The Unlock task is used to unlock specific locked resources.

To view the properties of an Unlock task:

Step 1  Create or retrieve the required workflow template.

Step 2  Choose the required unlock task in the drawing area.
Step 3  
Right-click the task, then choose **Task Properties**. The Task Properties dialog box for the required task is displayed (Figure 9-15).

*Figure 9-15  General Properties Tab - Unlock*

![General Properties Tab](image)

**Note**  
The Earliest Start option is not supported.

Step 4  
Choose the Attributes tab. The attributes of the task are displayed.
Step 5 Choose the Resource Names tab (Figure 9-16).

Figure 9-16 Resource Names Tab - Unlock

Step 6 In the text area, enter the resource names to be unlocked. Enter each resource name on a separate line. Resource names can be passed as parameters that are resolved during task execution to either workflow or task attributes:

- $Attribute name$—Evaluated from a workflow attribute by that name.
- $Task name:Attribute name$—Evaluated by a task attribute by that name.

Step 7 Click OK. The Task Properties dialog box is closed.

Testing a Workflow

You can test a workflow locally on the client.

To test a workflow:

Step 1 Create a new workflow in the Prime Network Workflow window (see Creating a New Workflow Template, page 9-4) or retrieve a workflow (see Retrieving a Workflow Template, page 9-5).

Step 2 In the toolbar, click Execute Workflow. A Workflow State Analysis tab that reflects the activation status is displayed in the Prime Network Workflow window. For more information about activation status, see Workflow Task, page 8-2.

The following views are displayed in the window and relate to workflow activation:

- Graphical View—Displays the workflow as a graphical presentation.
- Tabular View—Displays a table of all tasks in the workflow.
- Workflow State Analysis—Displays an analysis of the activation status of the workflow.
- XML View—Displays the XML code of the current workflow.

The following buttons are displayed at the bottom of the window:

- Activate—Activates the workflow.
Note: This button toggles to Abort when Activation is clicked. Click **Abort** to stop the workflow.

- Copy—Copies the workflow.
- Delete—Deletes a running workflow.

**Step 3** Click **Activate**. The Console window is displayed (Figure 9-17).

**Figure 9-17  Console Window**

The Console window displays the results of each task in the workflow.

**Step 4** Close the Console window. The Prime Network Workflow window is displayed.

The tasks displayed in the Graphical View tab (lower row) display the activation status of each task using an oval shape and colors (in the bottom right corner), as shown in Figure 9-18.

**Figure 9-18  Tasks Displayed In the Graphical View Tab**

The colors change as the status of the task changes during the process, as follows:
- Blue—Ready
- Green—Active
- Red—Abort
- Grey—Done
- Light Grey—Passive

For more information about activation status, see **Workflow Task, page 8-2**.

**Step 5** Review the results of running the workflow in the Prime Network Workflow window.
Once you have tested the workflow successfully on the local client, you can deploy the workflow to the gateway. For more information, see Deploying a Workflow Template, page 9-5.

Gateway Workflow Commands and Operations

This section describes how to add utility functions using BQL commands.

Deleting a Template

To delete a template from the server, use the following BQL command:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<command name="Delete">
  <param name="oid">
    <value>{[WorkflowTemplate(Name=template-name)]}</value>
  </param>
</command>
```

Running a Workflow

To execute a workflow based on an installed template, use the following BQL command:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<command name="RunWorkflow">
  <param name="templateOid">
    <value>{[WorkflowTemplate(Name=test.template)]}</value>
  </param>
  <param name="workflowAttributes">
    <value>
      <IWorkflowStringAttribute>
        <ID type="Oid">{[WorkflowAttribute(Name=param1)]}</ID>
        <Value type="String">name</Value>
      </IWorkflowStringAttribute>
      <IWorkflowIntegerAttribute>
        <ID type="Oid">{[WorkflowAttribute(Name=param2)]}</ID>
        <Value type="Integer">3</Value>
      </IWorkflowIntegerAttribute>
    </value>
  </param>
  <param name="preview">
    <value>false</value>
  </param>
</command>
```

…and similarly for other attributes…

```xml
<value>
  <IObject_Array/>
  <param name="preview">
    <value>false</value>
  </param>
</command>
```
If the value of preview is `true`, it will not execute any BQL tasks in the workflow that runs commands on a device. It will only provide a preview of the workflow BQL tasks that run the command on the device. The command in the BQL task will be executed only if the value of preview is `false`.

This command returns the OID of the new workflow; for example:

```
-- com.sheer.framework.imo.Oid --
{"Workflow(Id=801)}
```

### Aborting a Workflow

To abort a workflow, which includes stopping it if it is running and rolling back all activation scripts that it has executed, use the following BQL command:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<command name="AbortWorkflow">
  <param name="oid">
    <value>"{Workflow(Id=workflow-id)}"</value>
  </param>
</command>
```

If successful, this command returns nothing.

### Deleting a Workflow

To delete a workflow that is not active (that is, one that has already been completed or aborted), use the following BQL command:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<command name="Delete">
  <param name="oid">
    <value>{Workflow(Id=workflow-id)}"</value>
  </param>
</command>
```

If successful, this command returns nothing.

### Getting Workflow Output

To get the output of a workflow, both during its testing and afterward, use the following BQL command:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<command name="GetWorkflowOutput">
  <param name="oid">
    <value>{{Workflow(Id=workflow-id)}}"</value>
  </param>
</command>
```
This command returns the output as a string; for example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<IMO>
    <Output type="String">
        ...
    </Output>
    ...
</IMO>
```

[progress messages and script testing results]
Managing Workflows

Prime Network Administration can be used to manage deployed workflow templates.
Topics include:
- Starting Prime Network Administration, page 10-1
- Viewing the List of Templates and Template Properties, page 10-2
- Deleting a Workflow Template, page 10-2
- Viewing the Output of a Workflow, page 10-2
- Aborting a Workflow, page 10-3
- Deleting a Workflow, page 10-4
- Viewing the Workflow History Using Prime Network Events, page 10-4

Starting Prime Network Administration

This section provides instructions for launching Prime Network Administration. Prime Network Administration is password-protected to ensure security. Before you start, make sure you know the username, password, and Prime Network gateway IP address.

To start Prime Network Administration:

**Step 1** Choose Start > Programs > Cisco Prime Network >Cisco Prime Network Administration. The Cisco Prime Network Administration Login dialog box is displayed.

**Step 2** Enter your username, password, and host (Prime Network gateway IP address).

*Note* The gateway IP address that was used when you last logged in is automatically displayed in the Host field.

**Step 3** Click OK. The Cisco Prime Network Administration window is displayed.

*Note* For a detailed description of the Prime Network Administration application, see the Cisco Prime Network 3.8 Administrator Guide.
# Viewing the List of Templates and Template Properties

To view the list of templates and template properties:

**Step 1** In Prime Network Administration, choose **Workflow Engine > Templates**. The list of workflow templates is displayed in the table.

**Step 2** Choose the required workflow template in the table.

**Step 3** Right-click the template, then choose **Properties**. The Workflow Template Properties dialog box is displayed with the properties and attributes of the selected workflow template.

The name of the template is displayed in the header and at the top of the dialog box. The following property is displayed in the table in the Workflow Template Properties dialog box:

- **Name**—The attribute names defined for the tasks included in the workflow, as defined in the Task Properties dialog box using Prime Network Workflow.

# Deleting a Workflow Template

To delete a workflow template:

**Step 1** In Prime Network Administration, choose **Workflow Engine > Templates**. The list of workflow templates is displayed in the table.

**Step 2** Choose the workflow template that you want to delete in the table.

**Step 3** Delete the template in one of the following ways:

- Right-click the workflow template, then choose **Delete**.
- Click **Delete** in the toolbar.

A warning message is displayed.

**Step 4** Click **Yes**. The selected workflow template is deleted and no longer appears in the table.

# Viewing the Output of a Workflow

You can view the output of a workflow whether it is running, done, or aborted.

To view the output of a workflow:

**Step 1** In Prime Network Administration, choose **Workflow Engine > Workflows**.

**Step 2** Use the workflow search facility to locate the required workflow(s).

**Step 3** Right-click the workflow, then choose **Show Output**. The Output window is displayed (Figure 10-1).
Chapter 10  Managing Workflows

Aborting a Workflow

You can abort a workflow that is running. In addition, if any task in the workflow aborts, the workflow itself aborts.

When a workflow aborts, a workflow rollback occurs:

- Workflow rollback causes the activation scripts that have already been run (by Execute BQL task) to roll back.
- The rollback of an activation script is the execution of the rollback section of the script, as defined in Command Builder.
- Scripts roll back in the reverse order of their execution.

Gateway commands do not support rollback.

Rollback can be disabled for specific BQL tasks by setting the RollbackEnabled value task attribute to false in the respective BQL task. This is useful for a BQL task executing a script that does not have an appropriate rollback, or a BQL task executing a gateway command.

A workflow instance can only be aborted when it is running. It cannot be aborted when the process is done.

To abort a running workflow:

Step 4  Click Close. The Output window is closed.
Deleting a Workflow

To delete a workflow:

**Step 1** In Prime Network Administration, choose **Workflow Engine > Workflows**.

**Step 2** Use the workflow search facility to locate the required workflow(s).

**Step 3** Right-click the workflow, then choose **Abort**. A warning message is displayed.

**Step 4** Click **Yes**. The workflow is stopped, and the state of the workflow changes to **Aborted**.

Viewing the Workflow History Using Prime Network Events

Prime Network Events enables you to view the workflow history, including when workflows have been completed, their status, and the command invoked, as shown in Figure 10-2. The Provisioning Tab shows workflow events.

**Note**

For Prime Network Events to display an event when a Workflow is aborted, you must add the following line to the preActivateScript in the Callback Scripts tab:

```javascript
thisWorkflow.getRootTask().abort();
```

For more information, see Prime Network Workflow application online help for **Engine Behavior > Workflow State Transitions**.
You can also view the properties of the event, as shown in Figure 10-3.
The Description area of the Provisioning Event Properties window displays the details of the execution of the workflow, including all scripts, script rollback (if it occurred), and log messages.
Customizing Prime Network Workflow

This chapter explains how to extend the Workflow Engine with custom tasks and Prime Network Workflow callbacks.

Topics include:
- Coding, page 11-1
- Packaging for Deployment, page 11-3
- Deploying, page 11-4

Coding

This section includes the following topics:
- Custom Tasks, page 11-1
- Custom Task Panel Factories, page 11-2
- Workflow Callbacks Class, page 11-2

Custom Tasks

The procedure for developing custom tasks is explained in the Autonomy documentation.

The following is the definition of a simple task class:

```java
package samples;

import com.dralasoft.workflow.Key;
import com.dralasoft.workflow.SynchronousTask;

public class CustomTask1 extends SynchronousTask {
    public CustomTask1(Key _key) {
        super(_key);
    }

    public void perform() {
        System.err.println("Hello from CustomTask1");
    }
}
```
Custom Task Panel Factories

A TaskPanelFactory implementation class can be assigned to each custom task class. This allows you to create custom property sheets for the custom class. This factory class should implement the interface `com.dralasoft.gui.common.ext.TaskPanelFactory`. A simple way to do this is to extend `com.dralasoft.gui.common.ext.DefaultPanelFactory` and override some of its methods.

Workflow Callbacks Class

It is possible to develop a class that implements the Prime Network Workflow callbacks. This class should implement the interface `com.sheer.client.workflowstudio.IWorkflowEditorCallbacks`, which includes the following methods:

```java
/**
 * Called before the deploy template action
 * @param templateName
 * @return true if deploy template action should proceed
 */
public boolean preDeployTemplate(String templateName);

/**
 * Called after the deploy template action
 * @param templateName
 */
public void postDeployTemplate(String templateName);

/**
 * Called before the delete template action
 * @param templateName
 * @return true if delete template action should proceed
 */
public boolean preDeleteTemplate(String templateName);

/**
 * Called after the delete template action
 * @param templateName
 */
public void postDeleteTemplate(String templateName);

/**
 * @param templateName
 * @return true if this template should be displayed, false if not
 */
public boolean shouldDisplayTemplate(String templateName);
```
Packaging for Deployment

Classes and resources must be packaged into JAR files to be deployed on the server. A JAR file can contain multiple tasks and an optional workflow callback implementation.

In addition to the class files, each JAR file must contain a descriptor file named `extension-config.xml`. This file contains the XML block or blocks that define the tasks’ appearance in the task palette, the tasks’ custom panel factories, and, optionally, the Prime Network Workflow callback implementation.

Custom Tasks

In addition to the custom tasks class files, task icons should be included in the JAR file. Task icons should measure 16 by 16 pixels and must be placed in the JAR file in the subdirectory, com/dralasoft/gui/common/images/16x16.

Custom Task Panel Factories

The full class name of the task panel factory should be added in an element called `task-panel-factory-class` inside the custom-task element. If this element is not specified, `com.dralasoft.gui.common.ext.DefaultPanelFactory` is used for the custom task.

Workflow Callbacks Class

Adding an editor-callbacks-class element to the extension-config element configures a class that implements the Prime Network Workflow callbacks. This element should be added to the `extension-config.xml` file in only one of the JAR files (if the element is present in more then one descriptor, one element’s value would be used arbitrarily).

The following example of the contents of a task descriptor file defines two task types, one of which has a task panel factory, and an editor callback implementation:

```xml
<extension-config>
  <editor-callbacks-class>
    samples.EditorCallbacksImpl
  </editor-callbacks-class>

  <custom-task>
    <class-name>samples.CustomTask1</class-name>
    <label>Custom Task 1</label>
    <icon>task1.png</icon>
    <tooltip>Custom Task 1</tooltip>
    <menu-display>true</menu-display>
    <toolbar-display>true</toolbar-display>
    <task-panel-factory-class>
      com.sheer.client.workflowstudio.TestTaskPanelFactory
    </task-panel-factory-class>
  </custom-task>

  <custom-task>
    <class-name>samples.CustomTask2</class-name>
    <label>Custom Task 2</label>
    <icon>task2.png</icon>
    <tooltip>Custom Task 2</tooltip>
    <menu-display>true</menu-display>
  </custom-task>
</extension-config>
```
Deploying

To deploy JAR files:

**Step 1**
On the Prime Network gateway, copy the JAR files to the `/export/home/network38/dralasoft_extensions` directory.

*Note* Create the Prime Network gateway if necessary.

**Step 2**
Run the script `/export/home/network38/Main/scripts/installDralasoftExtensions.pl`.

*Note* Run this script each time you want to add a JAR file, remove a JAR file, or replace an existing JAR file with a new version.

The script installs and uninstalls JAR files so that the set of installed JAR files is the same as the contents of the directory `/export/home/network38/dralasoft_extensions`. We therefore recommend that you keep files with a `.jar` extension here even after they have been deployed.

*Note* The script restarts the Workflow engine, so we recommend that you confirm that no workflows are currently running in the engine.

**Step 3**
When the script is done, run Prime Network Workflow again. The deployed JAR files are automatically downloaded, and the new tasks are included in the task toolbar.

**About Workflow Migration**

If you are upgrading from an earlier version of Cisco ANA to Prime Network 3.8, see the installation guide for that version.

There is no change to the workflow engine in Prime Network 3.8, and the supported workflow version remains 3.6.6. Prime Network workflow templates are stored in the database and are available after upgrading to Prime Network 3.8.

Existing workflow templates that execute BQL inventory commands might require a review to verify the correctness and validity of these commands. Review the IMO and BQL command changes as described in the *Cisco Prime Network Integration Developer Guide* and in the IMO documentation, available on Cisco Prime Network Technology Center.
Your existing Prime Network Workflow AVM (avm66.xml) is maintained during the upgrade to Prime Network 3.8 so that any user-defined entries or customizations are available in Prime Network 3.8.

Any workflow extensions stored under ~/dralasoft_extensions are maintained during the upgrade to Prime Network 3.8.

Note

The workflow template must be saved or imported using the Export and Import options available in Prime Network Workflow. Saving or copying by any other method might not work as desired.
This part contains the following chapters:

- Chapter 12, “Introducing Command Builder”
- Chapter 13, “Working with Command Builder”
- Chapter 14, “Prime Network Macro Language”
- Chapter 15, “Creating a Prime Network Macro Language Command: An Example”
- Chapter 16, “BeanShell Commands”
Introducing Command Builder

Command Builder enables you to create new script and activation commands. Command scripts enable you to execute a programmable sequence of SNMP or Telnet command lines. These commands can include data properties taken from the Prime Network information model (built-in), as well as user-defined input parameters entered during runtime.

Because Command Builder is built into the standard Prime Network system platform:

- It utilizes regular Prime Network access interfaces to the network.
- Commands can be associated with any existing object group (IMO), type, or instance. This associated item is the working object on which the command is developed and tested.
- It enables access to the live information of the network object with which the command is associated.

Command Builder enables you to create commands using two different languages:

- Prime Network Macro Language—Uses a GUI wizard. For information about Prime Network Macro Language scripts, see Chapter 14, “Prime Network Macro Language”
- BeanShell scripting language.

Once defined, the commands are executed in one of the following ways:

- Interactively—Via autogenerated GUI forms (Prime Network Vision standard menus).
- Flow-through—Using the Prime Network API to integrate with external configuration applications.

A command can be published once it has been completed, enabling a wider scope of managed elements and network elements to use it. If you publish commands to multiple network elements, you can run them from the GUI on several network elements at once. Command results from all of the network elements are displayed in the command output.

Note

Changes to the registry should be performed only with the support of Cisco. For details, contact your Cisco account representative.

In addition to allowing you to create new commands, Prime Network provides a set of built-in, technology-based commands. You can run these commands but you cannot edit, publish or delete them. For details about supported commands, see the Cisco Prime Network 3.8 Reference Guide.

Additional information about Command Builder is available on the Cisco Developer Network (CDN).
About Command Builder Migration

Prime Network 3.8 supports command scripts created in Cisco ANA 3.6.x and Cisco ANA 3.7.x. Existing Command Builder commands written to Prime Network registry files (site*.xml or avm*.xml) are available after upgrading to Prime Network 3.8. If your command implementation includes references to IMOs or to the Prime Network internal model, it might require an update if there have been IMO changes.
WORKING WITH COMMAND BUILDER

This chapter describes how to use Command Builder to create, execute, and publish commands. Topics include:

- Opening Command Builder, page 13-1
- Command Builder Window, page 13-2
- A Workflow to Define a New Command, page 13-6
- Creating a Command, page 13-7
- Managing Commands, page 13-18

Opening Command Builder

This section provides instructions for launching Command Builder. Command Builder is launched from a specific managed element, which could be a managed element or a selected object within a managed element, such as a port. This managed element is used to develop and test the command. Once the command has been completed, it can be published and attached to a wider scope of managed elements.

To open Command Builder:

**Step 1** In Prime Network Vision, select the desired element:
- Right-click a managed element in the tree pane or context panel.
- Open the Inventory window for the required managed element, then right-click the required VNE inventory item.

**Note** For more information about the Inventory window, see the *Cisco Prime Network 3.8 User Guide*.

**Step 2** Choose Management > Command Builder. The Command Builder window is displayed.
Command Builder Window

Figure 13-1 shows an example of the Command Builder window.

Figure 13-1    Command Builder Window

The Command Builder window consists of:
- Table of Commands, page 13-2
- Menu Options, page 13-4
- Toolbar, page 13-4

Table of Commands

The Command Builder window displays a table of commands that are available for the selected managed element or network element. Command Builder enables you to:
- Add a new command, which enables you to execute a programmable sequence of Telnet or SNMP command lines.
- Edit an existing command.
- Delete a command that has not been published.
- Import and export command definitions.
- Import an updated version of an existing command definition.
- Test or run a command on the selected managed element.
- Publish a command and attach it to a wider scope of managed elements.
Table 13-1 describes the fields in the Command Builder table.

**Table 13-1  Command Builder Table**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The command name. This name is unique in the entire system.</td>
</tr>
<tr>
<td>Menu Caption</td>
<td>The text that is displayed in the menu when launching the command.</td>
</tr>
<tr>
<td>IMO Context</td>
<td>The inventory object associated with this command.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> A command is always associated with a selected object within a managed element, which enables it to use the properties of this object inside the script lines. For example, if you select a port object, the port’s properties, such as portAlias and status, are automatically made available to the script.</td>
</tr>
<tr>
<td>Local</td>
<td>Specifies whether the command is inherited from a higher level or is defined locally on the selected managed element. A command that is defined for a scope of managed elements (such as “All devices” or a specific device type) is automatically assigned to all managed elements in that scope. When modifications are made to a command that is inherited from a higher level, a local copy of the command is created for the specific managed element and overrides the generic definition. Once the local copy is tested and accepted, it can be published to update the higher-level definition.</td>
</tr>
</tbody>
</table>

A table can be sorted:

- According to a column by clicking the required column heading. The ✐ icon is displayed next to the selected column heading.
- In ascending or descending order by clicking the column heading. A triangle displayed next to the selected column heading indicates the sort order.

Clicking a red triangle displayed in a cell expands the cell to display all information in the cell.

A bar below the table displays the number of selected rows and the total number of rows in the table, such as 2/16 Selected. In addition, it displays the location of the selected rows in the table, such as Line 3.
Menu Options

Table 13-2 describes the menu options available in the Command Builder window.

Table 13-2  Command Builder Window Menu Options

<table>
<thead>
<tr>
<th>Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Menu</td>
<td></td>
</tr>
<tr>
<td>New Element</td>
<td>Creates a new command definition. For more information, see Creating a Command, page 13-7.</td>
</tr>
<tr>
<td>Edit Element</td>
<td>Edits an existing command definition. For more information, see Editing Commands, page 13-26.</td>
</tr>
<tr>
<td>Delete Element</td>
<td>Deletes an existing locally defined command; that is, a command that has not yet been published. For more information, see Deleting Commands, page 13-30.</td>
</tr>
<tr>
<td>Tools Menu</td>
<td></td>
</tr>
<tr>
<td>Export Element</td>
<td>Saves a full command definition that can later be imported to another managed element. For more information, see Exporting Commands, page 13-28.</td>
</tr>
<tr>
<td>Import Element</td>
<td>Performs one of the following actions, depending upon whether or not a version of the command definition already exists in Command Builder:</td>
</tr>
<tr>
<td></td>
<td>• For new command definitions, imports a full command definition to a managed element. For more information, see Importing Commands, page 13-28.</td>
</tr>
<tr>
<td></td>
<td>• For existing command definitions, replaces the existing command definition with an updated version of the command definition. For more information, see Updating Commands, page 13-29.</td>
</tr>
<tr>
<td>Hierarchy Manager</td>
<td>Moves the command definition to a different location or changes the scope of the command across the network hierarchy. For more information, see Publishing Commands, page 13-24.</td>
</tr>
<tr>
<td>Run Command</td>
<td>Previews or executes the command. For more information, see Executing Commands, page 13-18.</td>
</tr>
</tbody>
</table>

Toolbar

Table 13-3 identifies the buttons that appear in the Command Builder window.

Table 13-3  Command Builder Window Icons

<table>
<thead>
<tr>
<th>Button</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![New Element Icon]</td>
<td>New Element</td>
<td>Creates a new command definition. For more information, see Creating a Command, page 13-7.</td>
</tr>
<tr>
<td>![Edit Element Icon]</td>
<td>Edit Element</td>
<td>Edits an existing command definition. For more information, see Editing Commands, page 13-26.</td>
</tr>
</tbody>
</table>
Table 13-3 Command Builder Window Icons (continued)

<table>
<thead>
<tr>
<th>Button</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="icon" /></td>
<td>Delete Element</td>
<td>Deletes a command that exist locally; that is, a command that has not yet been published. For more information, see Deleting Commands, page 13-30.</td>
</tr>
<tr>
<td><img src="image2" alt="icon" /></td>
<td>Export Element</td>
<td>Saves a full command definition to a file that can later be imported to another managed element. For more information, see Exporting Commands, page 13-28.</td>
</tr>
</tbody>
</table>
| ![icon](image3) | Import Element     | 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, page 13-28.  
  - For existing command definitions, replaces the existing command definition with an updated version of the command definition. For more information, see Updating Commands, page 13-29. |
| ![icon4] | Hierarchy Manager  | Moves the command definition to a different location or changes the scope of the command across the network hierarchy. For more information, see Publishing Commands, page 13-24. |
| ![icon5] | Run Command        | Previews or executes a command. For more information, see Executing Commands, page 13-18. |
A Workflow to Define a New Command

Figure 13-2 illustrates the steps required to define a new command definition using Command Builder and the order in which the steps must be performed.

Figure 13-2 Define a New Command Workflow

- Create or edit a command
- Define the security access roles
- Define the input parameters
- Define the tab pages
- Define the script lines
- Save the 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.

Note

After you perform any action on the command (such as create, modify, or update), 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.

For more information about:

- Creating a command, see Creating a Command, page 13-7.
- Defining security access roles, defining input parameters, and defining tab pages, see Creating a Command, page 13-7.
- Defining script lines and saving commands, see:
  - Defining BeanShell Script Lines, page 13-16.
Creating a Command

Command Builder enables you to create a command definition which, by default, is created as a local instance. For more information about publishing a local instance of a command to a higher level in the hierarchy, see Publishing Commands, page 13-24.

In addition, you can edit an existing command. For information about editing an existing command, see Editing Commands, page 13-26.

To create a command:

**Step 1** In the Command Builder window, do one of the following:
- Click **New Element** in the toolbar.
- Choose **File > New Element**.

The New Command dialog box is displayed (Figure 13-3).

![New Command Dialog Box](image)

**Step 2** Enter the command identification information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>A unique name that identifies the command.</td>
</tr>
<tr>
<td>Menu Caption</td>
<td>Text that describes the command in the menu; that is, the menu option.</td>
</tr>
</tbody>
</table>
### Creating a Command

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Menu Visible</strong></td>
<td>Indicates whether or not the command is to appear as a menu option in Prime Network Vision. Check the check box for the command to appear as a menu option. Uncheck the check box to prevent the command from appearing as a menu option. <strong>Note</strong> A command that is defined as not visible in the menu can still be executed via the API.</td>
</tr>
<tr>
<td><strong>Menu Path</strong></td>
<td>If you check the Menu Visible check box, enter the location in the menu where the command is to be displayed.</td>
</tr>
<tr>
<td><strong>Context IMO</strong></td>
<td>The inventory object that is associated with the command (and exposes its data properties to the command). If the inventory object has subobjects that do not appear in the inventory tree (such as parameter groups of a port), they are listed in a drop-down list.</td>
</tr>
<tr>
<td><strong>Timeout</strong></td>
<td>The timeout value for the command, in milliseconds. The default value is 120000 milliseconds (2 minutes). <strong>Note</strong> If a timeout occurs, the script continues to run in the background. Upon its completion, a provisioning event is generated and can be viewed in Prime Network Events.</td>
</tr>
<tr>
<td><strong>Language</strong></td>
<td>The scripting language to be used for the command:</td>
</tr>
<tr>
<td></td>
<td>• Prime Network Macro—Simple mode. (See Chapter 14, “Prime Network Macro Language”.).</td>
</tr>
<tr>
<td></td>
<td>• BeanShell—Programmable mode; full scripting language. (See Chapter 14, “Prime Network Macro Language”.).</td>
</tr>
<tr>
<td><strong>Protocol</strong></td>
<td>The protocol to use:</td>
</tr>
<tr>
<td></td>
<td>• Telnet—Supported by both Prime Network Macro and BeanShell.</td>
</tr>
<tr>
<td></td>
<td>• SNMP—Supported by BeanShell only. This option does not appear if you choose Prime Network Macro in the Language field. <strong>Note</strong> Prime Network provides an SNMP BeanShell template that enables easy creation of SNMP commands.</td>
</tr>
</tbody>
</table>
Step 3  Click Next. The Command Authorizations dialog box is displayed.

![Command Authorizations Dialog Box](image)

Step 4  Select the security access roles that are authorized to execute the command:

- Administrator
- Configurator
- Operator Plus
- Operator
- Viewer

Step 5  Click Next. The User Input Arguments dialog box is displayed.

Command Builder enables you to define any number of input parameters. Input parameter attributes determine the structure and format of the input form. When the command is executed, the input form is generated automatically.

Note  The order of the input parameters determines the sequence in which they are presented in the input form.

Prime Network Macro Language supports two types of script 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 information about Prime Network Macro Language scripts, see Defining Prime Network Macro Language Script Lines, page 13-14.
Step 6 Click **Next** in the Command Authorizations dialog box. The User Input Arguments dialog box is displayed (Figure 13-5).

*Figure 13-5 User Input Arguments Dialog Box*

![User Input Arguments Dialog Box](image)

Step 7 To add a new argument, click **New**. The Add/Edit User Argument dialog box is displayed (Figure 13-6).

*Figure 13-6 Add/Edit User Argument for Command Dialog Box*

![Add/Edit User Argument for Command Dialog Box](image)
Step 8 Enter the required information:

<table>
<thead>
<tr>
<th>Property</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Parameter name. This entry must be unique and can contain only letters, numbers, hyphens (-), and underscores (_).</td>
</tr>
<tr>
<td>Caption</td>
<td>Parameter display name. This entry is displayed in the Command Builder script execution window.</td>
</tr>
<tr>
<td>Type</td>
<td>The type of input value:</td>
</tr>
<tr>
<td></td>
<td>• String</td>
</tr>
<tr>
<td></td>
<td>• Integer</td>
</tr>
<tr>
<td></td>
<td>• IPSubnet</td>
</tr>
<tr>
<td></td>
<td>• Combo</td>
</tr>
<tr>
<td></td>
<td>• IP Address</td>
</tr>
<tr>
<td></td>
<td>• Float</td>
</tr>
<tr>
<td></td>
<td>• Long</td>
</tr>
</tbody>
</table>

For more information about defining a Combo field type, see Defining a Combo Field Type, page 13-13.

**Note** Only input values that are valid for the selected type are accepted. These values are validated during runtime.

<table>
<thead>
<tr>
<th>Width</th>
<th>Field width, in number of characters. Relevant for the Command Builder script execution window.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible</td>
<td>Indicates whether this parameter appears in the Command Builder script execution window.</td>
</tr>
<tr>
<td></td>
<td>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).</td>
</tr>
</tbody>
</table>

**Note** When the parameter is not visible and has been assigned a default value, it can serve as a constant argument.

| Tooltip   | Displays the tooltip for the command parameter. This string is displayed for the parameter field in the input form (see Executing Commands, page 13-18). 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 Executing Commands, page 13-18). |

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 specify the sequence in which multiple arguments are presented when they are executed, select an argument and click **Move Up** or **Move Down**.

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

You can arrange the parameters in different tabs. By default, all of the parameters are displayed in a single tab, General.

**Figure 13-7 Tab Pages Dialog Box**
Step 12 Click New. The Add/Edit User Tab Page dialog box is displayed.

**Figure 13-8 Add/Edit User Tab Page for Command Dialog Box**

![Add/Edit User Tab Page for Command Dialog Box](image)

Step 13 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.

If you want different tabs for each of the parameters, repeat Steps 12 and 13.

Step 14 Click Next to define the command:

- For Prime Network Macro Language commands, see Defining Prime Network Macro Language Script Lines, page 13-14.
- For BeanShell commands, see Defining BeanShell Script Lines, page 13-16.

---

**Defining a Combo Field Type**

When Combo is selected in the Type field of the Add/Edit User Argument for x Command dialog box, the Browse button is enabled. This enables you to create a selection list (drop-down list) of the valid options that are displayed in the combo box of the input form, such as Up = 1 and Down = 2.

To define the combo entries:

**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 (Figure 13-9).
Creating a Command

Step 3 Enter the required information:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>The actual value of the option; for example, 1.</td>
</tr>
<tr>
<td>Label</td>
<td>The description of the entry that is displayed in the selection list (drop-down list) of the input form, such as Up.</td>
</tr>
</tbody>
</table>

Step 4 Click Add.
Step 5 Repeat Steps 3 and 4 as needed until you have added all entries.
Step 6 Click Close. The Add/Edit User Argument dialog box is displayed.

Defining Prime Network Macro Language Script Lines

You can specify either Prime Network Macro Language or BeanShell as the script language. Prime Network Macro Language scripts consist of a simple sequence of Telnet commands, runtime-replaced user-defined input parameters, and inline execution directives that are executed sequentially as Telnet configuration commands on a networking device.

Prime Network Macro Language represents both types of parameters (built-in and user-defined) 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 $vrfName$.
To define Prime Network Macro Language script lines (continuing from Step 14 in Creating a Command):

**Step 1** In the User Input Arguments dialog box, click Next. The Script Lines dialog box is displayed, enabling you to add or edit a script line, as shown in Figure 13-10.

**Figure 13-10  Script Lines Dialog Box**

![Script Lines Dialog Box](image)

**Step 2** 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script</td>
<td>The actual Telnet script lines sent to the device. The script lines can contain optional inline directives (pragmas) for finer granularity control. For more information about the supported pragmas, see Supported Pragmas, page 14-4.</td>
</tr>
</tbody>
</table>
Creating a Command

Step 3
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.

**Field** | **Description**
--- | ---
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
Click **Close** when the command has been successfully saved. The newly created command is displayed in the Command Builder table.

See Executing Commands, page 13-18 to preview or execute the command.

See Publishing Commands, page 13-24 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.

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.

### Defining BeanShell Script Lines

You can define the language of the script as either Prime Network Macro Language or BeanShell. BeanShell uses a fully programmatic logic via scripting language (including conditions, loops, and external files).

For information about Prime Network Macro Language scripts, see Defining Prime Network Macro Language Script Lines, page 13-14.
To define BeanShell script lines (continuing from Step 14 in Creating a Command):

**Step 1** In the User Input Arguments dialog box, click **Next**. The BeanShell Script dialog box is displayed.

**Step 2** 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.

- Unlike Prime Network Macro Language, in BeanShell user arguments, do **not** embed inventory properties within dollar signs ($...$).

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script</td>
<td>The actual Telnet script lines sent to the device. The script lines can contain optional inline directives (pragmas) for finer granularity control. For more information about the supported pragmas, see <a href="#">Supported Pragmas</a>, page 14-4.</td>
</tr>
<tr>
<td>Rollback</td>
<td>(Optional) The rollback script that is used when the command fails. <strong>Note</strong> If the rollback script fails, no additional actions can be performed.</td>
</tr>
</tbody>
</table>

**Step 3** Click **OK**. The selected field is displayed in the BeanShell Script dialog box.

**Step 4** 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.

For more information, see [Defining Prime Network Macro Language Script Lines](#), page 13-14.

**Step 5** Click **Close** when the command has been saved successfully. The newly created command is displayed in the Command Builder table.

See [Executing Commands](#), page 13-18 to preview or execute the command.

See [Publishing Commands](#), page 13-24 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.

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.
Managing Commands

Prime Network enables you to manage commands in the following ways:

- Preview or execute a command. See Executing Commands, page 13-18.
- Export a command for use with other network elements. See Exporting Commands, page 13-28.
- Import an updated version of a command. See Updating Commands, page 13-29.
- Delete a command. See Deleting Commands, page 13-30.

**Note**

We recommend that the users who manage Command Builder have the following:

- For their user account, the Configurator default permissions.
- For all assigned device scopes, the Configurator security level.

Creating users with a combination of security access roles lesser than Configurator may have security implications.

The above note is not applicable to executing commands because it depends on the roles you specify when you create the command.

Executing Commands

Command Builder enables you to execute a command immediately or to schedule the execution. Command Builder supports multiple activations, meaning that a command can be invoked to run concurrently on multiple managed elements or network elements. You can select multiple managed elements and execute a command that is shared with all the managed elements. If one of the network elements has a local copy of the command, the command is not listed when you select multiple managed elements. You must delete your local copy of the command after it is published. See Publishing Commands, page 13-24 for more details.

**Note**

Scheduling is not supported if multiple network elements are selected.

This section provides the following procedures for executing a command:

- Executing a Command Immediately, page 13-18
- Scheduling Command Execution, page 13-22
- Managing Command Jobs, page 13-23

### Executing a Command Immediately

To execute a command immediately:
Step 1  Select the required command in the Command Builder table and do one of the following:

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

**Note**  You can also run a command by right-clicking on a managed element and selecting the required command from the Commands menu.

The input form is generated and displayed (Figure 13-11). The title bar displays the name of the command.

**Figure 13-11  Input Form**

The input form contains the following tabs:

- **General**—Displays the input parameters required for the command.
- **Result**—Displays the preview or the actual interaction of the command in the output console with full execution audit.
- **Scheduling**—Allows you to schedule the command execution for a specified single or recurring time/date.
Step 2  Enter the relevant data in the input form.
Step 3  (Optional) Click **Preview** to see the command script, including variables, before it is executed.

**Note** DWDM and SONET show commands are executed when you click the Preview button. Instead of getting displayed in the Preview window, these commands are executed and the results are displayed. Configuration commands, however, work as expected.

Step 4  Click **Execute Now** to run the command immediately. The results of the command execution are displayed in the Result tab as shown in **Figure 13-12**.

**Figure 13-12  Command Results**

![Command Results Image]

Step 5  Close the input form to return to the Command Builder window.

**Note** You can also execute the command immediately from the Scheduling tab by selecting the Execute Now radio button and clicking **Execute Now**.
Scheduling Command Execution

To schedule execution of a command for a later date/time:

**Step 1** Select the required command in the Command Builder table and do one of the following:
- Click **Run Command** in the toolbar.
- Choose **Tools > Run Command**.
- Right-click the command, then choose **Run Command**.

**Note** You can also run a command by right-clicking on a managed element and selecting the required command from the Commands menu.

**Step 2** Enter the relevant data in the input form.

**Step 3** In the Scheduling tab, select the Schedule Job radio button.
Step 4 For a single job execution, select the Once radio button and specify the date and time the command should be run.

For a recurring command execution, select the Recurring radio button and specify the following:

- The time range for the recurrence.
- How often you want the command to be executed within that time range - every X minutes, daily, weekly, or monthly.

Step 5 Click Schedule Job. The command job will be created and the command will be executed according to your scheduling specifications.

Step 6 To check that your command job has been created, select Tools > Command Jobs.

Managing Command Jobs

When you schedule the execution of a command, a job is created. Command jobs can be viewed and managed in the Command Jobs window.

To view and manage command jobs:

Step 1 Select Tools > Command Jobs. The Command Jobs window displays a list of command jobs.

Figure 13-14 Command Jobs

The Command Jobs window displays the following information for each job:

- Job Name—The unique name of the job, derived from the command name.
- Next Run—The time and date the job is scheduled to run next.
- Owner—Which user created the job.
- Status—The current status of the job (Scheduled, Completed, Suspended, Canceled).
- Last Run—The last time and date the job was run.
- Last Run Status—The status of the last job (Unknown, Completed, Partial Success, Failed, Rolled Back).
- Comments—Displays the comments entered by the user when scheduling the command execution.
Step 2  Select a job in the list. The toolbar and right-click menu enable you to perform the following actions for the selected job:

- View job properties, including the command results script.
- Delete the job
- Refresh the job window to see the latest status of the job.
- Stop the job (when it is running).
- Pause the job (when it is running).
- Resume the job (when it has been paused or stopped).
- View a list of all executions of this job (for recurring jobs). This list shows the date and time the job was run and the status of each job run.

Publishing Commands

After you have defined a command and tested it on a specific instance of a managed element, you can publish it and apply it to a wider scope of managed elements in the network. You can then select multiple network elements and run a command on all of them at the same time.

The Command Builder Hierarchy Manager dialog box enables you to publish the command to one or more locations across the inheritance hierarchy (as defined in the system). In other words, you define the scope where the command is to be applied in the hierarchy.

Different variations of a command can be used for different managed elements and network elements, where the implementation of the command is different for each managed element or network element. Figure 13-15 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 13-15  Inheritance Hierarchy Example**

![Inheritance Hierarchy Example](image)

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 defined at a higher level and is assigned to all devices of type Cisco 7200 in the system.

**Note**  A user with a Configurator role can add and publish commands on all devices regardless of the device scope defined for the user.
To publish a command:

**Step 1** Select the required command in the Command Builder table.

**Step 2** Open the Hierarchy Manager dialog box in one of the following ways:

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

**Note** If user-friendly VNE names exist in the schema, the hierarchy manager table displays these user-friendly registry location names in the VNE Hierarchy Location column. A user-friendly VNE name is a hierarchy path that has been defined in the registry and is then displayed in the hierarchy manager table.

Each row that is displayed in the Hierarchy Manager dialog box represents a different level of the hierarchy. The 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 displayed in the table:

- **Exist**—A check mark in this column 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 13-4 describes the tools that are displayed in the Hierarchy Manager dialog box.

<table>
<thead>
<tr>
<th><strong>Table 13-4</strong></th>
<th><strong>Hierarchy Manager Dialog Box Tools</strong></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>Button</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Copy Icon" /></td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="Cut Icon" /></td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="Paste Icon" /></td>
<td>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.</td>
</tr>
<tr>
<td><img src="image" alt="Delete Icon" /></td>
<td>Deletes the command from the selected node in the hierarchy.</td>
</tr>
</tbody>
</table>

**Note** If the command has been deleted from all nodes, it is removed from the list in the Command Builder window.
Managing Commands

Chapter 13 Working with Command Builder

Managing Commands

Step 3 Select the node in the hierarchy from where you want to Copy/Cut the command.
Step 4 Click Copy or Cut in the toolbar to copy or cut the command.
Step 5 Select the node in the hierarchy where you want to publish the command.
Step 6 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.

Editing Commands

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.

Editing Commands

You can edit an existing command definition created by you. The command that is edited affects only the local instance. When an inherited command is edited, the new local instance overrides the generic command definition for the specific managed element.

When you edit a command, the command that is edited will be saved as a local instance. If a local instance does not exist, it will be created. In the case of a published command, if you want the edits to reflect in the inheritance hierarchy, you must re-publish the local instance.

Table 13-4 Hierarchy Manager Dialog Box Tools (continued)

<table>
<thead>
<tr>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![File Icon]</td>
<td>Saves a full command definition to a file that can later be imported to another managed element. For more information, see Exporting Commands, page 13-28.</td>
</tr>
</tbody>
</table>
| ![Cut Icon] | 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, page 13-28.
  - For existing command definitions, replaces the existing command definition with an updated version of the command definition. For more information, see Updating Commands, page 13-29. |
To edit a command:

**Step 1** Select the command that you want to edit, and then do one of the following:
- Click **Edit Element** in the toolbar.
- Choose **File > Edit Element**.
- Right-click the command, then choose **Edit Element**.

The Hierarchy Manager dialog box is displayed. See **Editing Predefined Commands** for information about editing predefined commands.

**Note** If user-friendly VNE names exist in the schema, the hierarchy manager table displays these user-friendly registry location names in the VNE Hierarchy Location column. A user-friendly VNE name is a hierarchy path that has been defined in the registry and is then displayed in the hierarchy manager table.

**Step 2** Select the required version of the command from the hierarchy manager, then click **Next**. The Edit Command dialog box is displayed for the selected command.

**Step 3** Edit the command as required. For more information, see **Creating a Command**, page 13-7.

**Step 4** Click **Finish** in the script dialog box to save the modified command.

**Step 5** When the command has been successfully saved, click **Close**. The edited command is supported and displayed in the Command Builder window.

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.

For information about previewing and executing the edited command, see **Executing Commands**, page 13-18.

For information about publishing the edited command, see **Publishing Commands**, page 13-24.

**Editing Predefined Commands**

Prime Network supports editing and customizing predefined commands.

When you edit a predefined command, Command Builder creates a clone, as a local copy. You can rename the predefined command when you edit it. This copy of the predefined command can then be modified, published, or exported like other user-defined commands.

To edit a predefined command:

**Step 1** Select the predefined command that you want to edit, and then do one of the following:
- Click **Edit Element** in the toolbar.
- Choose **File > Edit Element**.
- Right-click the command, then choose **Edit Element**.

The Edit Command- Command Identification dialog box is displayed. The command name and the menu caption are displayed as:
Managing Commands

You can enter a new command name and menu caption.

**Step 2** Enter a command name and caption.

**Step 3** Edit the command as required. For more information, see Creating a Command, page 13-7.

**Step 4** Click **Finish** in the script dialog box to save the modified command.

**Step 5** When the command has been successfully saved, click **Close**.

Exporting Commands

To export commands:

**Step 1** In the Command Builder window, select the command that you want to export. You can export only one command at a time.

**Step 2** Open the Export dialog box in one of the following ways:
- Click **Export Element** in the toolbar.
- Choose **Tools** > **Export Element**.
- Right-click the command, then choose **Export Element**.

**Step 3** Select the command that you want to export by checking the check box in the Selected column. The version is selected in the table.

**Step 4** Click **OK**. The Export Property dialog box is displayed.

**Step 5** Browse to the directory where you want to save the command.

**Step 6** In the File name field, enter a name and extension (for example, .txt) for the command.

**Step 7** Click **Save**. The command is saved in the selected directory. The Export dialog box is displayed.

**Step 8** Click **Close**. The Command Builder window is displayed.

Importing Commands

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 Prime Network Vision, select the required managed element.

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

**Step 3** Open the Import Element dialog box in one of the following ways:
- Click **Import Element** in the toolbar.
- Choose **Tools** > **Import Element**.
Managing Commands

Step 4  Browse to the directory that contains the commands that you want to import.
Step 5  Select the commands that you want to import. To select multiple commands, press Shift or Ctrl while choosing the commands.
Step 6  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 7  In the Import Element dialog box, select the VNE hierarchy location for the specified command.
Step 8  Click OK. The Command Builder window is displayed.
Step 9  Click Close. The commands are imported and displayed in the Command Builder window.

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.

Updating Commands

You can import an updated version of a command. When you import an updated version of a command that already exists in Command Builder, you get an error message which states that the command already exists. If you want to import the command, you must delete the existing command instance. When you import an updated version of the command, the existing version in Command Builder is replaced with the new command.

Note  After you 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 updated version from the registry.

To update a command:

Step 1  In Prime Network Vision, select the required managed element.
Step 2  Right-click the element, then choose Management > Command Builder. The Command Builder window is displayed.
Step 3  Open the Import Element dialog box in one of the following ways:
  •  Click Import Element in the toolbar.
  •  Choose Tools > Import Element.
Step 4  Browse to the directory that contains the updated version of the command definition.
Step 5  Select the updated command definition that you want to import. You can import only one updated command definition at a time.
Step 6  Click Open. The Import Elements dialog box is displayed.
Step 7  In the Import Elements dialog box, select the VNE hierarchy location for the updated command definition.
Step 8  Click OK. A window appears, asking you to confirm the operation.
Step 9  Click Yes to confirm the operation, or No to cancel the operation.
If you click Yes, the updated version of the command definition replaces the existing version of the command definition, and the Command Builder window is displayed.

**Step 10** Click Close to close the Command Builder window.

---

### Reviewing Command History

Every command that is executed is logged in the Prime Network event database. The command’s execution history can be viewed using the Prime Network Events application. For more information about Prime Network Events, see the *Cisco Prime Network 3.8 User Guide*.

### Deleting Commands

By default, a command created by a user is created as a local instance. A command that is defined locally can be seen in the Command Builder window. You can only delete commands that exist locally; that is, commands that have not yet been published.

You can delete a user-created command directly from Command Builder only if the command has a single local instance. In other words, only commands that are not published and exist locally can be deleted from Command Builder. If the command has more than one instance, you must go to Hierarchy Manager and delete the instances from the required nodes. You can delete all instances, if required.

You can delete multiple commands at a time as long as all of the commands are associated with the same VNE.

To delete commands:

**Step 1** In the Command Builder window, select the commands you want to delete. To select multiple commands, press **Shift** or **Ctrl** while choosing the commands.

**Step 2** Do one of the following:

- Click **Delete Element** in the toolbar.
- Choose **File > Delete Element**.
- Right-click one of the selected commands, then choose **Delete Element**.

**Step 3** When prompted, confirm or cancel the deletion process.

If you confirm the deletion, the commands are deleted and are no longer displayed in the Command Builder table.
Prime Network Macro Language

This chapter describes 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. This chapter is intended for use by programmers who want to write command scripts that are executed within the Prime Network activation framework.

Topics include:
- What Are Prime Network Macro Language Scripts?, page 14-1
- Properties Available from the IMO Context, page 14-2
- Specifying and Using Parameters, page 14-2
- Supported Pragmas, page 14-4
- Example, page 14-9

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 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:

- Managed device
  - IMO name—IManagedElement
  - Example properties—CommunicationStateEnum, DeviceName, ElementType
- Port
  - IMO name—IPortConnector
  - Example properties—portalias, location, ifindex

For more information about Prime Network Macro Language Built-in parameters, see Built-In Parameters, page 14-4.

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 Builder GUI, 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 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).

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 Builder or through the Prime Network API.

Table 14-1 provides a complete list of user-defined parameter properties.

### Table 14-1 Available User-Defined Parameters

<table>
<thead>
<tr>
<th>Property</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Parameter name. Can contain only letters, digits, hyphen (-), and underscore (_), and must be unique.</td>
</tr>
<tr>
<td>Caption</td>
<td>Parameter display name. Visible in the Command Builder script execution window.</td>
</tr>
<tr>
<td>Type</td>
<td>String, Integer, IPSubnet, Combo, IP, Float, Long.</td>
</tr>
<tr>
<td>Width</td>
<td>Field width, in characters. Relevant for the Command Builder script execution window.</td>
</tr>
<tr>
<td>Visible</td>
<td>Indicates whether or not the parameter appears in the window. Relevant for the Command Builder script execution window.</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Tooltip for the command parameter.</td>
</tr>
<tr>
<td>Default</td>
<td>A default value for the parameter.</td>
</tr>
<tr>
<td>Required</td>
<td>Indicates whether the argument is mandatory or optional.</td>
</tr>
</tbody>
</table>

**Note**

Some parameter properties are relevant only for the script data entry window in 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 14-2, using the example 198.168.2.10 255.255.255.0.

Table 14-2  Formats for IP Subnet Parameters

<table>
<thead>
<tr>
<th>#</th>
<th>Format</th>
<th>Description</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>maskbits</td>
<td>The IP of the subnet converted to an integer value. Bits only.</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>ip</td>
<td>Only the IP without the mask.</td>
<td>198.168.2.10</td>
</tr>
<tr>
<td>3</td>
<td>mask</td>
<td>The IP of the subnet mask without the IP address.</td>
<td>255.255.255.0</td>
</tr>
<tr>
<td>4</td>
<td>networkmask</td>
<td>The mask address converted to the network.</td>
<td>0.0.0.255</td>
</tr>
<tr>
<td>5</td>
<td>ipmaskbits</td>
<td>The IP and the value of the mask bits.</td>
<td>IP/30</td>
</tr>
<tr>
<td>6</td>
<td>ipmask</td>
<td>The IP mask. This is the default.</td>
<td>198.168.2.10 255.255.255.0</td>
</tr>
<tr>
<td>7</td>
<td>ipmasknot</td>
<td>The IP and the network address.</td>
<td>198.168.2.10 + 0.0.0.255</td>
</tr>
</tbody>
</table>

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.

For a complete list of the available built-in parameters related to the IMO context, see the Cisco Prime Network 3.8 Reference Guide.

Note: To view all user-defined and built-in parameters in the 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 14-3 lists the pragmas that Prime Network Macro Language scripts support.

Table 14-3  Supported Pragmas

<table>
<thead>
<tr>
<th>Pragma</th>
<th>Short Description</th>
<th>Refer to…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success</td>
<td>Line-specific success check.</td>
<td>Success</td>
</tr>
<tr>
<td>Fail</td>
<td>Line-specific failure check.</td>
<td>Fail</td>
</tr>
</tbody>
</table>
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`. 

---

### Supported Pragmas (continued)

<table>
<thead>
<tr>
<th>Pragma</th>
<th>Short Description</th>
<th>Refer to…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt</td>
<td>Line-specific prompt assertion validation.</td>
<td>Prompt</td>
</tr>
<tr>
<td>Full prompt</td>
<td>Full prompt, line-specific prompt assertion validation.</td>
<td>Full Prompt</td>
</tr>
<tr>
<td>Rollback</td>
<td>Rollback enable or disable.</td>
<td>Rollback</td>
</tr>
<tr>
<td>Activity</td>
<td>Script remarks. These also help determine the failure location.</td>
<td>Activity</td>
</tr>
<tr>
<td>Enum</td>
<td>Defining enumerated value substitution.</td>
<td>Enum</td>
</tr>
</tbody>
</table>

---
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 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 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 `<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.
Supported Pragmas

Example

The following example changes the Telnet prompt and validates the change in the newly returned Telnet prompt:

```plaintext
configure terminal [prompt={config}]
```

This pragma succeeds only if the next device prompt ends with `{config)#`.

Full Prompt

Description

A **full 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

```plaintext
[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 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:

```plaintext
configure terminal [prompt=^router(config)#]
```

This pragma yields success only if the next device prompt matches `router(config)#` exactly.

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 14-4 lists the user-defined argument definitions used in the script.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Default</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>vrfName</td>
<td>String</td>
<td>N/A</td>
<td>The VRF name. The value provided for this argument is used as the VRF table name.</td>
<td>Manhattan</td>
</tr>
<tr>
<td>rt</td>
<td>String</td>
<td>N/A</td>
<td>The VRF route target, in the format integer:integer. The value provided for this argument is used as is for the device configuration.</td>
<td>60:60</td>
</tr>
<tr>
<td>rd</td>
<td>String</td>
<td>1</td>
<td>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.</td>
<td>1, 2, or any valid value according to the enum pragma</td>
</tr>
</tbody>
</table>
Table 14-5 provides an explanation of the command script line by line.

**Table 14-5   Command Script Explanation**

<table>
<thead>
<tr>
<th>Script Line</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>enum rd 1=60:60;2=80:80</td>
<td>The line enumerates the possible values of the route distinguisher argument.</td>
</tr>
<tr>
<td>show ip vrf $vrfName$ [success=% No VRF named $vrfName$]</td>
<td>Verify if the requested VRF already exists. Continue to create the VRF only if the requested VRF name is not found.</td>
</tr>
<tr>
<td>[activity=prepare for VRF creation]</td>
<td>Remark to state that the following section is preparation for VRF creation.</td>
</tr>
<tr>
<td>config terminal [success=Enter configuration commands, one per line. End with CNTL/Z.] [prompt=(config)]</td>
<td>Change mode command. Continue to the next command if the success pragma string is found in the device reply and prompt changes to config.</td>
</tr>
<tr>
<td>ip vrf $vrfName$ [prompt=(config-vrf)]</td>
<td>Change mode command. Continue to the next command if prompt changes to config-vrf.</td>
</tr>
<tr>
<td>[rollback]</td>
<td>Placeholder to state that rollback should be executed only if a subsequent script line fails.</td>
</tr>
<tr>
<td>[activity=create VRF]</td>
<td>Remark to state that the following section is actually the VRF creation.</td>
</tr>
<tr>
<td>rd $rd$ [fail=% Cannot set RD, check if it's unique]</td>
<td>Set the route distinguisher. If this command fails, the rollback script is called.</td>
</tr>
<tr>
<td>route-target both $rt$</td>
<td>Set the route target. If this command fails, the rollback script is called.</td>
</tr>
<tr>
<td>end</td>
<td>Change mode command. Return to normal (enable) mode.</td>
</tr>
</tbody>
</table>

Table 14-6 provides an explanation of the activation rollback script line by line.

**Table 14-6   Rollback Script Explanation**

<table>
<thead>
<tr>
<th>Script Line</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>config terminal</td>
<td>Set the device to terminal mode.</td>
</tr>
<tr>
<td>no ip vrf $vrfName$</td>
<td>Delete the VRF from the device.</td>
</tr>
<tr>
<td>end</td>
<td>Change mode command. Return to normal (enable) mode.</td>
</tr>
</tbody>
</table>
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
Creating a Prime Network Macro Language Command: An Example

This chapter describes creating a Prime Network Macro Language addvrf command from start to finish. To create the addvrf command:

**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 15-1, enter the information for the parameters.

<table>
<thead>
<tr>
<th>Field</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
<th>Parameter 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>vrfname</td>
<td>rd</td>
<td>rt</td>
</tr>
<tr>
<td>Caption</td>
<td>VRF Name</td>
<td>Route Distinguisher</td>
<td>Route Target Both</td>
</tr>
</tbody>
</table>

*Table 15-1*  
addvrf Command Parameters
**Table 15-1  addvrf Command Parameters (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Parameter 1</th>
<th>Parameter 2</th>
<th>Parameter 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>String</td>
<td>String</td>
<td>String</td>
</tr>
<tr>
<td>Width</td>
<td>15</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>Visible</td>
<td>Select</td>
<td>Select</td>
<td>Select</td>
</tr>
<tr>
<td>Tooltip</td>
<td>Enter VRF name</td>
<td>Enter Route</td>
<td>Enter Target</td>
</tr>
<tr>
<td>Required</td>
<td>Checked</td>
<td>Unchecked</td>
<td>Checked</td>
</tr>
</tbody>
</table>

**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 15-1).

**Figure 15-1  User Input Arguments Dialog Box**

**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.
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 15-3.

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 the 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.
CHAPTER 16

BeanShell Commands

This chapter describes the methods that should be used for BeanShell in Prime Network commands when you want to interact with devices. 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 ($...$).

Topics include:
- Telnet Examples, page 16-2
- SNMP Examples, page 16-3

Figure 16-1 presents the methods that should be used for BeanShell in Prime Network commands when interacting with devices for Telnet and SNMP interfaces.

Figure 16-1 BeanShell Methods

<table>
<thead>
<tr>
<th>Device/TelnetInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>setCommand(string)</td>
</tr>
<tr>
<td>setStatus(int status)</td>
</tr>
<tr>
<td>setStatus(int status, string)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Device/SNMPInterface</th>
</tr>
</thead>
<tbody>
<tr>
<td>ifCommand(string)</td>
</tr>
<tr>
<td>isCommand(string)</td>
</tr>
<tr>
<td>getStatus(string)</td>
</tr>
<tr>
<td>$Command( int $status )</td>
</tr>
</tbody>
</table>

Note
For setStatus, 1 = success and 2 = failure.
Telnet Examples

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; used, for example, to signal to Workflow Manager that a configuration command has succeeded or failed.

- `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:

```java
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 Examples

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; used, for example, to signal Workflow Manager that a configuration command has succeeded or failed.

For additional information about general scripting language, see http://www.beanshell.org/.
PART 5

VNE Customization Builder (VCB)

This part contains the following chapters:
- Chapter 17, “Introducing the VNE Customization Builder (VCB)”
- Chapter 18, “Enabling Support for Additional Device Types and Software Versions”
- Chapter 19, “Enabling Support for Additional Modules”
- Chapter 20, “Managing Additional Event Types Using the VCB”
- Chapter 22, “VCB Template Reference”
CHAPTER 17

Introducing the VNE Customization Builder (VCB)

This chapter introduces the Virtual Network Element Customization Builder (VCB) in Cisco Prime Network.

Topics include:

- About the VCB, page 17-1
- A Quick Tour of the VCB, page 17-2
- Testing and Certifying VCB Customizations - Overview, page 17-4

Additional information about the VCB is available on the Cisco Developer Network (CDN).

About the VCB

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

The VCB can be used to:

- 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.

The VCB writes extensions into a local registry file, site.xml, thereby avoiding impact on the Prime Network code. You do not need to know a great deal about the Prime Network registry to use the VCB.
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).

The VCB tool in Prime Network Administration provides a simple and intuitive GUI for performing VCB functions. If you are new to the VCB, we recommend using this 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.

Note
The system supports a maximum of two simultaneous VCB user sessions.

The VCB procedures in this guide focus on the VCB GUI. VCB CLI commands are detailed in the following sections:

- Using the CLI to Create and Manage U-VNEs, page 18-19
- Using the CLI to Add and Manage Modules, page 19-9
- Using the CLI to Customize Events, page 20-13
- Chapter 22, “VCB Template Reference”.

VCB Task Flow

1. Plan—Research the NE, module, or event that you intend to add or modify
2. Define Customizations—Define your customizations using the VCB GUI or CLI, preferably in a lab or staging environment.
3. Test—Verify and certify your customizations using your own test resources, such as a simulator.
4. Move to Production—During a maintenance window, put tested, certified extensions into your production environment.

A Quick Tour of the VCB

Note
You must be an Administrator to use the VCB. For more information on user authorization, see the Cisco Prime Network 3.8 Administrator Guide.

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 tool:

Step 1
In Prime Network Administration, choose Tools > VNE Customization Builder.
In your web browser, enter the following URL: https://gateway IP address:8043/prime-network-web

The Cisco Prime Network Home page is displayed.

**Step 2** Click the VNE Customization Builder link under Tools in the Site Map area. The VCB opens, and the VNE Drivers page is displayed.

![Figure 17-1 VCB - VNE Drivers Page](image)

**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.

### Filtering the VCB Tables

The VCB tool provides a simple and an advanced filter.

To apply a simple filter:

**Step 1** In the VCB tool, click the **Filter** button or select **Quick Filter** from the drop-down list in the Show field above the table.

A free text field is displayed under each column heading and enables you to filter information in each column. The table is filtered as you type.

**Step 2** To clear the filter and perform a new search, click the **Filter** button.
To apply an advanced filter:

**Step 1** In the VCB tool, select Advanced Filter from the drop-down list in the Show field above the table.

**Step 2** Select the attribute, operator and conditions by which you want to filter the table.

**Step 3** To add another filter rule, click the "+" sign.

**Step 4** Click **Go** to apply the filter.

---

**Exporting VCB Tables**

The VCB tool allows you to export tables to a CSV file.

To export to CSV:

**Step 1** In the VCB tool, click the Export button in the tool bar. The system automatically gives the CSV file a name.

**Step 2** Choose whether you want to save or open the file and click OK.

---

**Note**

The export to CSV feature does not work on certain Internet Explorer browsers if the default Internet Explorer settings are used. This is because https content cannot be stored locally on the hard drive. To enable saving content locally, uncheck the “Do not save encrypted pages to disk” option in the Internet Explorer advanced Internet options.

---

**Testing and Certifying VCB Customizations - Overview**

Procedures for testing and certifying customizations are provided throughout this guide. 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

For more information, see **The VCB Log File, page 17-4** and **Test Resources, page 17-5**.

---

**The VCB Log File**

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.
Test Resources

To perform the necessary tests on VCB customizations, make use of the tools and resources listed in Table 17-1.

Note

Some tools listed in Table 17-1 are resources to consider in addition to Prime Network.

Table 17-1 Testing Tools and Resources

<table>
<thead>
<tr>
<th>Tool/Resource</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| Prime Network Vision client   | • Adding an instance of the U-VNE that you created with the VCB and verifying its capabilities.  
                                 | • Verifying the physical and logical inventory of the device.             
                                 | • Verifying that events from the device are being received.              
                                 | • Checking 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             | Chapter 22, “VCB Template Reference” provides the list of technologies and properties that are supported by the template on which the U-VNE is based.  
                                 | The Cisco Active Network Abstraction 3.7 Reference Guide provides the list of technologies and properties supported by the developed VNEs in a particular version of Prime Network. |
| Trap simulator (not supplied with Prime Network) | Enables testing of new events. |
Enabling Support for Additional Device Types and Software Versions

The chapter contains the following sections:

- Enabling Support for Additional Device Types by Creating U-VNEs, page 18-1
- Creating a U-VNE, page 18-8
- Managing Existing VNE Drivers, page 18-11
- Enabling Support for Additional Software Versions, page 18-16
- Using the CLI to Create and Manage U-VNEs, page 18-19
- Testing and Certifying U-VNEs, page 18-36

Enabling 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 user-defined VNE drivers, called U-VNEs. After you have created a U-VNE, you can add it to an AVM using Prime Network Administration, and 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.

This section contains the following subsections:

- Approaches to Creating a U-VNE, page 18-2
- How are U-VNEs Represented in Prime Network?, page 18-2
- Advantages and Limitations of U-VNEs, page 18-3
- Generic VNE, U-VNE, and Developed VNE Comparison, page 18-5
- Planning for Creating a U-VNE, page 18-9
- Creating a U-VNE - Procedure, page 18-10
- Testing and Certifying U-VNEs, page 18-36
- Moving a U-VNE to Production, page 18-44
Approaches to Creating a U-VNE

Several approaches are available for creating U-VNEs. Depending on the approach you choose, the modeling capabilities will be more specific or less specific. The available approaches are listed below from most specific to least specific:

1. Create a U-VNE by cloning a developed VNE driver. This approach will produce the most specific results in terms of device modelling. Use this approach when the unsupported device has the same or a very similar management interface, and/or device OS version, as the developed VNE driver.

   For example, a new device that belongs to a supported device series, or runs a new maintenance device OS release would typically share the same management interface as a supported device. A U-VNE created by cloning 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. (For supported device types and OS versions, see the Cisco Prime Network 3.8 Reference Guide.)

2. Create a U-VNE based on a supported device family. Use this approach when there is no specific developed VNE driver to clone but the unsupported device belongs to a supported device family.

3. Create a U-VNE based on a supported software version. Use this approach if there is no specific device to clone but you know that the unsupported device’s software version is supported by a developed VNE driver.

4. Create a U-VNE by referencing a U-VNE template. This approach will produce the least specific results in terms of device modelling. Use this approach when the instrumentation or management interface for the unsupported device does not match that of any supported VNE. Typically, non-Cisco devices and Cisco device families for which there are no developed VNE drivers would be in this category. See U-VNE Templates, page 22-1.

U-VNEs are created as separate VNE definitions in the Prime Network registry, and can be further extended using Prime Network soft-properties and command builder features.

How are U-VNEs Represented in Prime Network?

In the Prime Network GUI, all managed devices are represented by a unique name and an icon, as shown in the Prime Network Vision map below.
Chapter 18  Enabling Support for Additional Device Types and Software Versions

Enabling Support for Additional Device Types by Creating U-VNEs

Figure 18-1  Device Type Representation in Prime Network Vision Map

The icon represents the device category (Ethernet Switch, Router, and so on) or the device family (Cisco Catalyst 3750 Metro Series Switches, Cisco 12000 Series Routers, and so on) to which the device belongs.

When you create a U-VNE, you define its “device type”. This determines which name and icon will be used to represent the new device in the Prime Network maps and tables. The device type definition includes providing a unique name for the device, and specifying the device category and the device family to which the device belongs. See Defining Device Types to Determine the GUI Representation of VNE Drivers, page 18-13.

Advantages and Limitations of U-VNEs

U-VNEs enable you to manage additional device types and to manage additional traps and syslogs. However, U-VNEs are not as powerful as developed VNE drivers for the following reasons:

- Dependence on U-VNE templates—If the U-VNE references a template, it will support only the NE instrumentation for device discovery that is defined in the selected U-VNE template.
- Standard MIB NE instrumentation support—NE instrumentation is based on the 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.
- Limited fault management—Supports traps for U-VNEs that were created using the Generic U-VNE template; by design, syslogs are not supported by the Generic U-VNE template. Otherwise, developed VNEs and U-VNEs are presented with almost the same limitations and capabilities:
For events that were added via the VCB, correlation is done based on the DC key (correlation using network flow is not supported). The event is correlated to service events with the same source, but users cannot customize correlation parameters such as weight, correlation delay, and so on. In addition, events added via the VCB cannot be root cause events.

- For factory-defined events, capabilities are the same for developed VNEs and U-VNEs, provided that the U-VNE supports the technology.

- Limited path tracing—U-VNEs that were created using the Generic U-VNE template provide limited support for path tracing. If the U-VNE does not support routing, network paths that traverse the U-VNE will stop there. Therefore, the result of the trace will show only the path to or from the U-VNE. The exact behavior depends on the technologies used by the device, the level of adherence to standard MIB support, and the location of the U-VNE on the path. We recommend that you keep this limitation in mind when considering whether to use a U-VNE to manage a particular device, especially a device in the network core.

\[\text{Note}\]  
A U-VNE created via cloning will have the same capability as the device from which it was cloned.

- Limited topology support—U-VNEs support the Ethernet topology, including its underlying physical topology, on Ethernet ports. However, dynamic topology discovery is limited to U-VNEs on the network edge that are connected to a developed VNE. In addition, U-VNEs support IP topologies on high-level data link control (HDLC) and serial ports under the following conditions:
  - Point-to-point links
  - No duplicate IP addresses
  - Same IP subnet

- Limited trap support—U-VNE trap support is limited to those listed in the specific U-VNE template.

\[\text{Note}\]  
For a side-by-side comparison of generic VNEs, developed VNEs, and U-VNEs, see Generic VNE, U-VNE, and Developed VNE Comparison, page 18-5.
Generic VNE, U-VNE, and Developed VNE Comparison

Table 18-1 compares the features available in generic VNEs, U-VNEs, and developed VNEs. The information in Table 18-1 is for comparative purposes only. The support provided by each U-VNE template is on a best-effort basis and might vary from template to template.

**Table 18-1 Comparison of VNE Types**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Generic VNEs (using the VCB)</th>
<th>U-VNEs</th>
<th>Developed VNEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customization</td>
<td>Partial (events)</td>
<td>Yes</td>
<td>Yes (modules and events)</td>
</tr>
</tbody>
</table>
| Fault Analysis           | Support for standard traps that are supported by regular Generic VNEs, and can be extended to support proprietary traps and sylogs. Refer to the *Cisco Prime Network 3.8 Reference Guide* for a full list of supported traps. | Standard and proprietary traps and sylogs that are:  
  - Defined in the U-VNE template (when a U-VNE is added using a template)  
  - Supported by the device from which the U-VNE was cloned (when a U-VNE is added by cloning) | Standard and proprietary traps and sylogs |
| MIB support              | Standard MIB-II              | Standard MIB-II and private MIBs for:  
  - Physical inventory discovery, depending on the selected U-VNE template  
  - Importing traps to add event recognition | Standard MIB-II and private MIBs |
### Table 18-1 Comparison of VNE Types (continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Generic VNEs</th>
<th>U-VNEs</th>
<th>Developed VNEs</th>
</tr>
</thead>
</table>
| CLI or XML over Telnet is used to query NE | No           | • For a clone-based U-VNE, the protocol used to query is inherited from cloned-from VNE driver  
• For a template-based U-VNE, the protocol used to query depends on what is defined in the U-VNE template | Yes            |
| Multivendor                            | Yes          | Yes                                                                    | Yes            |
| NBI                                    | Yes          | Yes                                                                    | Yes            |
| Discovery                              | Yes          | Yes (CDP is supported for Cisco devices)                              | Yes            |
| Physical Inventory, Containment        | No           | Yes, depending on the selected template and the NE instrumentation     | 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 interfaces¹ | Depends on Prime Network supported technologies and NE instrumentation; see the *Cisco Prime Network 3.8 Reference Guide* |

¹ Depends on the NE instrumentation.
Table 18-1  Comparison of VNE Types (continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Generic VNEs</th>
<th>U-VNEs</th>
<th>Developed VNEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm Correlation</td>
<td>No</td>
<td>Yes for factory-defined events only, provided that the related</td>
<td>Yes (for factory-defined events)</td>
</tr>
<tr>
<td>(root cause analysis)</td>
<td></td>
<td>technologies are supported by the U-VNE²</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Event associated to managed element by default</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>  For events that were added via the VCB, correlation is done</td>
<td>For events that were added via the VCB, correlation is done based on the DC key</td>
</tr>
<tr>
<td></td>
<td></td>
<td>on the DC key (correlation using network flow is not supported).</td>
<td>(correlation using network flow is not supported). The event is correlated to service events with the same source, but</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The event is correlated to service events with the same source, but</td>
<td>users cannot customize correlation parameters such as weight, correlation delay, and so on. In addition, events added via the VCB cannot be root cause events.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>users cannot customize correlation parameters such as weight,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>correlation delay, and so on.</td>
<td></td>
</tr>
<tr>
<td>NE Image Management</td>
<td>No</td>
<td>No</td>
<td>Yes (for factory-defined events)</td>
</tr>
<tr>
<td>Configuration Archival</td>
<td>No</td>
<td>No</td>
<td>Yes (for factory-defined events)</td>
</tr>
<tr>
<td>Path Tracing</td>
<td>No</td>
<td>Only physical and Ethernet are supported</td>
<td>Yes (for factory-defined events)</td>
</tr>
<tr>
<td>Topology Discovery</td>
<td>Physical and</td>
<td>• For template-based U-VNEs, only physical and Ethernet are supported</td>
<td>Full (for factory-defined events)</td>
</tr>
<tr>
<td></td>
<td>Ethernet</td>
<td>— Supported technologies are documented in the templates</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• For cloned U-VNEs, depends on the VNE from which it was cloned¹</td>
<td></td>
</tr>
</tbody>
</table>

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.

Creating a U-VNE

The following sections describe the steps involved in creating a U-VNE:

- Creating a U-VNE—A Step-by-Step Example, page 18-8
- Planning for Creating a U-VNE, page 18-9
- Creating a U-VNE - Procedure, page 18-10
- Testing and Certifying U-VNEs, page 18-36

Creating a U-VNE—A Step-by-Step Example

The following example shows how to enable Prime Network to manage 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 generic U-VNE template.

**Step 1** Obtain the sysOID for the NE that you want to manage by querying the device using, for example, the snmpget command:

```
snmpget -c SNMPCommunityString IPAddress .1.3.6.1.2.1.1.2.0
```

**Step 2** Confirm that Prime Network does not already support the sysOID. Do one of the following:

- In the VCB GUI, filter the VNE Drivers table by the sysOID. If no driver is found, it is not supported. See Filtering the VCB Tables, page 17-3 for information about the filter.
- Search for the sysOID in the output of the following command on the Prime Network gateway:

```
vcb uvne view -sysoid all | grep sysOID
```

If the value that you entered for `sysOID` is not listed, it is not supported.

**Step 3** Decide on the approach you want to take to add the U-VNE. See Approaches to Creating a U-VNE, page 18-2. For this example, the new U-VNE will be based on the generic U-VNE template.

**Step 4** Open Prime Network Administration.

**Step 5** Click **Tools > VNE Drivers**.

The VNE Drivers tab lists all the VNE drivers.

**Step 6** Click **Add Row**. A row is added to the table, enabling you to enter values in the relevant fields or select from the drop-down lists.

**Step 7** Enter the sysOID of the new device in the SysOID field.

**Step 8** Specify the name that will represent the device in the Prime Network GUI by creating a new device type, as follows:

- In the Device Type field, click the down arrow to display the Device Type Selector.
- Click the Tools icon in the upper right area of the Device Type Selector and select **Add**.
- 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.
Chapter 18      Enabling Support for Additional Device Types and Software Versions

Creating a U-VNE

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

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

Step 9 In the Type field, select User Defined VNE - by Template.

Step 10 In the Cloning Reference field, select GenericUVNE.

Step 11 Click Save.

Step 12 Prepare to test; see Setting Up the Test Environment, page 18-37.

Step 13 Complete tests and certify the customization; see Procedures for Testing and Certifying U-VNEs, page 18-39.

Planning for Creating a U-VNE

Before you create a U-VNE, you need to have a good understanding of the device type for which you want to enable support in. When creating a U-VNE to enable support for an additional device type, you have several options, as described in Approaches to Creating a U-VNE, page 18-2. Before you can decide which approach to take, the initial steps of Creating a U-VNE—A Step-by-Step Example, page 18-8 describe obtaining the sysObjectID for a device and confirming that it is not supported in Prime Network before you start customizing. While you are in the planning phase, here are some additional steps to assist you in determining the approach to take when creating a U-VNE:

1. Perform research, investigating the NE that you want to add. Gather information from sources such as the following:
   - Vendor- or device-specific documentation or website for details.
   - Element management systems (EMS) or other management systems for property or detail views.
   - Device or network experts.
   - Advanced services or CA personnel who deal with Prime Network.
   - VCB commands that enable you to view details for similar VNEs, modules, and events—See Viewing Existing Device Types, page 18-14,
   - Cisco Prime Network 3.8 Reference Guide to compare the supported device scope with the device scope that you need to manage. If gaps exist—in the areas of recognizing device type, hard cards, device maintenance software, SNMP traps, or syslogs—use the VCB to extend the VNE drivers.

2. Evaluate whether you can clone. To create a U-VNE by cloning from an existing VNE driver, you must identify a suitable VNE driver.

   For the technologies that are modeled for a particular scheme under a particular device series, see the Cisco Prime Network 3.8 Reference Guide. After you identify the device series, use the vcb uvne view command to obtain the sysoid of a particular device type as follows:

   `vcb uvne view -sysoid all -user root -password admin`

   **Note** In the output of the command, ignore any entries for which the Device Family column contains base or da.
3. If the cloning approach is not a good fit for the device that you want to discover, add the U-VNE by referencing a template. Perform analysis, looking for gaps between the information that you gathered and the information in the templates; see U-VNE Templates, page 22-1. Evaluate the U-VNE templates to determine which template most closely matches the data that you gathered.

4. Obtain the sysOID for the NE that you want to manage by doing any of the following:
   - Query the device using, for example, the snmpget command
     \[\text{snmpget} \ -c \text{SNMPCommunityString} \text{IPAddress} \ .1.3.6.1.2.1.1.2.0\]
   - Use any MIB browser and query for the OID \(1.3.6.1.2.1.1.2.0\)
   - Look into the CISCO-PRODUCTS-MIB if you are investigating a Cisco product. The following URL opens the Cisco SNMP Object Navigator to the CISCO-PRODUCT-MIB:

5. Confirm that Prime Network does not already support the sysOID by doing one of the following:
   - Search for the sysOID in the output of the following command:
     \[\text{vcb uvne view -sysoid all | grep sysOID}\]
     If the value that you entered for \textit{sysOID} is not listed, it is not supported.

---

**Creating a U-VNE - Procedure**

To add a new VNE driver using the Prime Network VCB GUI:

**Step 1** Click **Tools > VNE Drivers**.

The VNE Drivers tab lists all the VNE drivers.

**Step 2** Click **Add Row**. A row is added to the table, enabling you to enter values in the relevant fields or select from the drop-down lists.

Define the following parameters for the new VNE driver:
Managing Existing VNE Drivers

The Prime Network VCB GUI enables you to manage VNE drivers in your network. You can perform the following actions to manage the VNE Driver:

- Editing a VNE Driver, page 18-12
- Deleting a VNE Driver, page 18-12

### Managing Existing VNE Drivers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SysOID</td>
<td>The unique object identifier of the VNE driver. For more details on how to obtain the SysOID for the U-VNE that you want to manage, see Planning for Creating a U-VNE, page 18-9</td>
</tr>
</tbody>
</table>
| Device Type     | The name that will represent the device in the Prime Network GUI. Click the down arrow in the Device Type field to display the Device Type Selector which contains a list of existing device types for selection. In the Device Type Selector, you can:  
  - Select a device type.  
  - Create a new device type by clicking on the Tools icon and then selecting Add. We recommend that you prefix the device type name with the network element vendor name for easy identification. The new device type will be added to the list and you will be able to select it.  
  - Toggle between the tree-structure view and the standard view.  
  - Click Back to go to the previous view. |
| Type            | The type of the VNE driver:  
  - Prime Network Default  
  - User-Defined VNE- by Template  
  - User-Defined VNE- by SysOID  
  - User-Defined VNE- by Software Version  
  - User-Defined VNE- by Device Family  
  You can use these options to define if you want to create a VNE driver based on device family or software version. See Approaches to Creating a U-VNE, page 18-2 for more information. |
| Cloning Reference | The item from which you want to clone the new VNE.                                                                                                                                                          |
| Overriding System Default | This parameter is populated automatically by the system:  
  Yes—If the U-VNE was created by editing a system default VNE.  
  No—If the U-VNE was created from scratch and not by editing a system default VNE. |

**Step 3** Click **Save**.

**Step 4** Test and certify the U-VNE. See Testing and Certifying U-VNEs, page 18-36.
Managing Existing VNE Drivers

Viewing Existing VNE Drivers

To view VNE drivers:

Step 1 In the VCB tool, click **Tools > VNE Drivers**.
For more details of the selected VNE driver, click the arrow icon in each row.

Editing a VNE Driver

To edit an existing row in the VNE Driver table, double-click or click on the row you want to edit.

To edit an existing VNE driver using the Prime Network VCB GUI:

Step 1 Log into the Prime Network VCB GUI using the login credentials.
On successful login, Prime Network VCB GUI displays the home page.

Step 2 Click **Tools > VNE Drivers**.
The VNE Drivers tab lists all the VNE drivers.

Step 3 Select a VNE driver from the list.

Step 4 Click **Edit**.

Note You will receive a warning message while trying to edit the Prime Network Default VNE driver. Click **Ok** to continue and **Cancel** to view the VNE Drivers window.

Step 5 Click **Save** to save the changes.

Step 6 Click **Cancel** to view the VNE Drivers window.

Step 7 Restart the AVM, that manages the devices.
Step 8 Test the VNEs.
Step 9 Restart the Prime Network server.

Deleting a VNE Driver

To delete an existing VNE driver using the Prime Network VCB GUI:
Step 1 Log into the Prime Network VCB GUI using the login credentials. On successful login, Prime Network VCB GUI displays the home page.

Step 2 Click Tools > VNE Drivers. The VNE Drivers tab lists all the VNE drivers.

Step 3 Select a VNE driver from the VNE Drivers list.

Step 4 Click Delete.

Note You cannot delete the default VNE Drivers.

Step 5 A window appears prompting you to confirm deletion of selected VNE driver. Click OK to continue deleting the device type.

Step 6 Click Cancel to view the VNE Drivers window.

Step 7 Restart the AVM, that manages the devices.

Step 8 Test the VNEs.

Step 9 Restart the Prime Network server.

Viewing Modules in a Selected VNE Driver

To view modules in an existing device type using the Prime Network VCB GUI:

Step 1 Log into the Prime Network VCB GUI using the login credentials. On successful login, Prime Network VCB GUI displays the home page.

Step 2 Click Tools > VNE Drivers. The VNE Drivers tab lists all the VNE drivers.

Step 3 Click the arrow to the left of the required VNE driver to expand its display.

Step 4 Click the Module Group link. The displayed dialog box lists all the modules in the module group.

Note For details on how to add new module and manage existing modules, see Chapter 19, “Enabling Support for Additional Modules”.

Defining Device Types to Determine the GUI Representation of VNE Drivers

You can perform the following actions to manage device types:

- Viewing Existing Device Types, page 18-14
- Adding a New Device Type, page 18-14
- Editing a Device Type, page 18-15
Viewing Existing Device Types

To view a list of device types:

Step 1  In the VCB window, click Tools > VNE Drivers.

Step 2  Click Configure Device Types. The Device Types window appears.

The Device Types window provides a table listing all the available device types. The table columns show the following information for each device type:

- **Device Type**—The name that represents the VNE driver in the Prime Network GUI.
- **Category**—The type of device the VNE driver represents. For example, Router, Eth-Switch, and so on.
- **Device Series**—The device family. For example, Cisco 12000 Series Routers.
- **Type**—Whether the device type is provided with the system (system default) or user-defined.

Adding a New Device Type

If none of the existing device types is a suitable representation of your U-VNE, you can create a new device type. You can add 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 - Procedure, page 18-10.
To add a new device type:

**Step 1** In the VCB window, click **Tools > VNE Drivers**.

**Step 2** Click **Configure Device Types**. The Device Types window appears.

**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:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Type</td>
<td>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.</td>
</tr>
<tr>
<td>Category</td>
<td>The type of device, for example, router, switch, and so on.</td>
</tr>
<tr>
<td>Device Series</td>
<td>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.</td>
</tr>
<tr>
<td>Type</td>
<td>Since you are adding a device type that is not provided as a system default, this field is automatically populated as “User-Defined”.</td>
</tr>
</tbody>
</table>

**Step 5** Click **Save**.

### Editing a Device Type

You can edit user-defined device types. System default device types cannot be edited.

To edit an existing device type:

**Step 1** In the VCB window, click **Tools > VNE Drivers**.

**Step 2** Click **Configure Device Types**. The Device Types window appears.

**Step 3** Double-click on a device type or select a device type and click **Edit**.

**Step 4** Change the device type parameters as required. See **Adding a New Device Type, page 18-14** for information about the parameters.

**Step 5** Click **Save**.
Deleting a Device Type

You can delete user-defined device types. System default device types cannot be deleted.

To delete a device type:

**Step 1** In the VCB window, click **Tools > VNE Drivers**.

**Step 2** Click **Configure Device Types**. The Device Types window appears.

**Step 3** Select a device type from the device types list.

**Step 4** Click **Delete**. You are prompted to confirm that you want to delete the device type. Click **OK** to continue deleting the device type.

Enabling Support for Additional Software Versions

The VCB enables you to enable support for additional software versions and to manage currently supported software versions. You can perform the following actions to manage the software version:

- Adding Support for a New Software Version, page 18-18
- Editing a Software Version, page 18-18
- Deleting a Software Version, page 18-19

Viewing the Software Versions

To view the software versions of a selected VNE driver in Prime Network VCB GUI:

**Step 1** Log into the Prime Network VCB GUI application.

**Step 2** Click **Tools > VNE Drivers**.

**Step 3** Click the arrow next to the required VNE driver to expand its display.
Figure 18-3  Accessing the Link to Supported Software Versions

Step 4  Click supported s/w versions in the Software Versions field.

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

Figure 18-4  Software Versions
You can perform the following actions from the Software Versions window:

- Edit a software version—Select a software version in the list and click Edit.
- Delete a software version—Select a software version in the list and click Delete.

## Adding Support for a New Software Version

To add support for a new software version:

**Step 1**
In the VCB window, click Tools > VNE Drivers.

**Step 2**
Click the arrow next to the VNE driver for which you want to add the new software version.

**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.

**Step 4**
Click Add Row.

**Step 5**
Enter the values for the following parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Software Version</td>
<td>The name of the software version to be supported.</td>
</tr>
<tr>
<td>Scheme</td>
<td>The scheme of the new VNE, either Product or IpCore.</td>
</tr>
<tr>
<td>Supported Software Version</td>
<td>The already supported software version to be cloned in order to add support for the new software version.</td>
</tr>
</tbody>
</table>

**Step 6**
Click OK.

## Editing a Software Version

*Note* You cannot edit system default software versions.

To edit a software version:

**Step 1**
In the VCB window, click Tools > VNE Drivers.

**Step 2**
Click the arrow next to the VNE driver for which you want to add the new software version.

**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.

**Step 4** Select the software version you want to edit.

**Step 5** Click **Edit**.

**Step 6** Edit the software version as required.

**Step 7** Click **OK**.

---

### Deleting a Software Version

**Note**
You cannot delete system default software versions.

To delete a software version:

**Step 1** In the VCB window, click **Tools > VNE Drivers**.

**Step 2** Click the arrow next to the VNE driver for which you want to add the new software version.

**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.

**Step 4** Select the software version you want to delete.

**Step 5** Click **Delete**.

**Step 6** Click **OK**. A confirmation dialog box is displayed.

**Step 7** Click **OK**.

---

### Using the CLI to Create and Manage U-VNEs

You have the option to use the CLI to perform VCB customizations. Use the CLI if you are an advanced VCB user or if the current VCB GUI does not support the customization you need to perform.

This section describes the CLI commands that can be used to create and manage U-VNEs and device types, as follows:

- Introductory Command Reference and Global Options, page 18-20
- Commands for Adding and Managing U-VNEs, page 18-23
- Commands for Adding and Managing Device Types, page 18-31
Introductory Command Reference and Global Options

This command reference introduces the **vcb** command and gives details for the **vcb sitechanges** command which you must use to deploy, maintain, or remove any VCB customizations. This command reference includes the following sections:

- vcb, page 18-20
- vcb sitechanges, page 18-22

**vcb**

The **vcb** command is the wrapper for all VCB-related commands.

**Synopsis**

The syntax of the **vcb** command is as follows:

```
{uvne|module|pluggablemodule|event|eventpattern|eventparsingrule|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:

- **Commands for Adding and Managing U-VNEs**, page 18-23
- **Commands for Adding and Managing Device Types**, page 18-31
- **Commands for Managing Modules**, page 19-10
- **Commands for Managing Pluggable Modules**, page 19-18
- **Using the CLI to Customize Events**, page 20-13
Global Command Options

Table 18-4 describes the global options and arguments that are common to all vcb commands and subcommands.

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>help</td>
<td></td>
<td>Displays online help about the command. Use this option for each subcommand.</td>
</tr>
</tbody>
</table>
| debuglevel |          | Determines which messages are logged based on the message severity. Valid values are:  
  • error  
  • warn (default)  
  • info  
  • 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.
Chapter 18  Enabling Support for Additional Device Types and Software Versions

General Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Signifies OK.</td>
</tr>
<tr>
<td>10</td>
<td>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.</td>
</tr>
<tr>
<td>11</td>
<td>User does not have Prime Network admin privileges.</td>
</tr>
<tr>
<td>20</td>
<td>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.</td>
</tr>
<tr>
<td>30</td>
<td>Failed to connect to Prime Network server.</td>
</tr>
<tr>
<td>40</td>
<td>I/O error. Some physical input or output error occurred.</td>
</tr>
<tr>
<td>50</td>
<td>Invalid argument. Some invalid argument was supplied.</td>
</tr>
<tr>
<td>60</td>
<td>Incompatible Prime Network version of VCB; expected $vcb_expected_version$ and found Prime Network $ana_running_version$.</td>
</tr>
<tr>
<td>70</td>
<td>Bad procedure for program. A Remote Procedure Call (RPC) call was attempted for a procedure which does not exist in the VCB program.</td>
</tr>
<tr>
<td>80</td>
<td>Function not implemented. Attempted a system call that is not available on this system.</td>
</tr>
</tbody>
</table>

**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.

**Synopsis**

```
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.
Commands for Adding and Managing U-VNEs

Use the following `vcb uvne` commands to create, view, modify, and delete U-VNEs:

- `vcb uvne add`, page 18-23
- `vcb uvne view`, page 18-25
- `vcb uvne modify`, page 18-28
- `vcb uvne delete`, page 18-30

**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.

**Synopsis**

```
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 18-31.

**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 18-31).
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 18-25.)

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), that points to registrations—scheme and instrumentation commands—of the already supported software version 12.0(23)S2 for the device family 100xx. (For a list of already supported software versions, see vcb uvne view, page 18-25.)

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 18-6 Options and Arguments—vcb uvne add

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysoid</td>
<td>sysoid</td>
<td>The sysObject ID of the device for which to create a U-VNE-driver using the implementation defined in the U-VNE template.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> Each sysOID value in the system must be unique.</td>
</tr>
<tr>
<td>template</td>
<td>template name</td>
<td>The name of the U-VNE template from which to create the U-VNE-driver. See U-VNE Templates, page 22-1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> This option is mutually exclusive with the -clonesysoid and -clonedevicefamily options.</td>
</tr>
<tr>
<td>group</td>
<td>template filename</td>
<td>The name of the file in which the U-VNE template is located. U-VNE templates are located in the uvne-product file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> Use this option with the -template option only.</td>
</tr>
<tr>
<td>devicetype</td>
<td>device type name</td>
<td>(Optional) The U-VNE devicetype name. If not specified, the device type is:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Defined as Unknown when the option is not specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Inherited from the device or device family when you use the -clonesysoid or -clonedevicefamily options.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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 18-31.</td>
</tr>
<tr>
<td>clonesysoid</td>
<td>clone-sysoid</td>
<td>The sysObject ID of an already supported VNE. To view sysOIDs for supported VNEs, use the vcb uvne view -sysoid all command.</td>
</tr>
</tbody>
</table>
Chapter 18  Enabling Support for Additional Device Types and Software Versions

Using the CLI to Create and Manage U-VNEs

Table 18-6  Options and Arguments—vcb uvne add (continued)

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>softwareversion</td>
<td>The software version that you want to add for a U-VNE. Obtained using the show version command.</td>
</tr>
<tr>
<td>clone-softwaredversion</td>
<td>An already supported software version that you want to clone. Software version must already be supported.</td>
</tr>
<tr>
<td>clone-softwareversion</td>
<td>For a list of supported device families, use the command vcb uvne view –scheme &lt;ipcore</td>
</tr>
<tr>
<td>cloneddevicefamily</td>
<td>A supported device family. For a list of supported device families, use the command vcb uvne view –scheme &lt;ipcore</td>
</tr>
<tr>
<td>devicefamily</td>
<td>Note: This option is mutually exclusive with the -template option.</td>
</tr>
</tbody>
</table>

Table 18-7  Error Codes—vcb uvne add

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>The sysOID already exists and is already modeled as a VNE.</td>
</tr>
<tr>
<td>102</td>
<td>U-VNE template file not found.</td>
</tr>
<tr>
<td>103</td>
<td>No such template name in the templates file.</td>
</tr>
<tr>
<td>104</td>
<td>No such device type configuration exists.</td>
</tr>
</tbody>
</table>

Note: For the list of general VCB error codes, see General Error Codes, page 18-22.

vcb uvne view

The vcb uvne view command returns an existing U-VNE configuration.

Synopsis

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
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 18-8 Options and Arguments—vcb uvne view

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysoid</td>
<td>sysoid</td>
<td>Returns the configuration of the specified VNE, including the device type,</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the user-friendly name, and the template (for U-VNEs).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tip</strong> Enter <strong>all</strong> as the <strong>sysoid</strong> to view the configuration of all</td>
</tr>
<tr>
<td></td>
<td></td>
<td>sysOIDs (U-VNEs and developed VNEs) configured in the system.</td>
</tr>
<tr>
<td>group</td>
<td>template filename</td>
<td>Returns the configuration of all sysOIDs and templates contained in the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>specified group.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tip</strong> Enter <strong>all</strong> as the <strong>template filename</strong> to view the template</td>
</tr>
<tr>
<td></td>
<td></td>
<td>associations for each group in the system.</td>
</tr>
<tr>
<td>userdefined</td>
<td></td>
<td>Lists the U-Vne created through vcb command only.</td>
</tr>
<tr>
<td>details</td>
<td></td>
<td>Shows following additional information:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Module Spec Name associated with the sysoid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Syslog and Trap Parsing rule name associated with the sysoid</td>
</tr>
</tbody>
</table>

Note: For the list of global options, see Global Command Options, page 18-21.

Error Codes

Table 18-9 Error Codes—vcb uvne view

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>U-VNE template file not found.</td>
</tr>
<tr>
<td>103</td>
<td>No such template name in the templates file.</td>
</tr>
<tr>
<td>104</td>
<td>No such device type configuration exists.</td>
</tr>
<tr>
<td>111</td>
<td>The SysOID specified does not exist.</td>
</tr>
</tbody>
</table>

Note: For the list of general VCB error codes, see General Error Codes, page 18-22.
**vcb uvne modify**

The `vcb uvne modify` command modifies the configuration of an existing U-VNE.

**Synopsis**

```
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.

**Usage Examples**

**Example 1**

```
vcb uvne modify -sysoid .1.2.3.4 -group uvne-product -template cisco-chassis-mib-template -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 -softwareversion 12.6(2) -clonesoftwareversion 12.0(23)S3 -clonedevicefamily 100xx -scheme product -user root -password admin
```

Modifies support for a device with sysOID .1.2.3.4, to point to a new software version, 12.0(23)S3, for the device family 100xx.

**Example 5**

```
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 18-10  Options and Arguments—vcb uvne modify

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysoid</td>
<td>sysoid</td>
<td>The sysObject ID of the U-VNE-driver configuration that you want to modify.</td>
</tr>
<tr>
<td>template</td>
<td>template name</td>
<td>The name of the U-VNE template to which the U-VNE should be associated. Use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>this option to modify the U-VNE template from which the U-VNE-driver derives</td>
</tr>
<tr>
<td></td>
<td></td>
<td>its configuration.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> Using this option does not overwrite other configuration changes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>made with the VCB, such as user-experience attributes that are defined with</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the <code>vcb devicetype add</code> command.</td>
</tr>
<tr>
<td>group</td>
<td>template filename</td>
<td>The name of the group that includes the U-VNE template, such as uvne-product.</td>
</tr>
<tr>
<td>devicetype</td>
<td>device type</td>
<td>The device type associated with the U-VNE.</td>
</tr>
<tr>
<td>clonesysoid</td>
<td>clone-sysoid</td>
<td>The sysObject ID of an already supported VNE. To view sysOIDs for supported</td>
</tr>
<tr>
<td></td>
<td></td>
<td>VNEs, use the <code>vcb uvne -sysoid</code> command.</td>
</tr>
<tr>
<td>softwareversion</td>
<td>software-version</td>
<td>The software version that you want to support for a U-VNE. To obtain the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>software version string, use the <code>show version</code> command.</td>
</tr>
<tr>
<td>clonessoftwareversion</td>
<td>clone-softwareversion</td>
<td>An already supported software version that you want to clone. The software</td>
</tr>
<tr>
<td></td>
<td></td>
<td>version must already be supported for a particular device family.</td>
</tr>
<tr>
<td>clonedevicefamily</td>
<td>deviceFamily</td>
<td>A supported device family. For a list of supported device families, use</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the command <code>vcb uvne -sysoid</code> all --user user -password password`. For</td>
</tr>
<tr>
<td></td>
<td></td>
<td>more information, see <code>vcb uvne view</code>, page 18-25.</td>
</tr>
</tbody>
</table>

**Note** For the list of global options, see Global Command Options, page 18-21.

Error Codes

Table 18-11  Error Codes—vcb uvne modify

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>102</td>
<td>U-VNE template file not found.</td>
</tr>
<tr>
<td>103</td>
<td>No such template name in the templates file.</td>
</tr>
<tr>
<td>104</td>
<td>No such device type configuration exists.</td>
</tr>
<tr>
<td>112</td>
<td>The sysOID does not exist or already exists and</td>
</tr>
<tr>
<td></td>
<td>is already modeled as a VNE.</td>
</tr>
</tbody>
</table>

**Note** For the list of general VCB error codes, see General Error Codes, page 18-22.
The `vcb uvne delete` command deletes a U-VNE.

**Synopsis**

```
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
```

Delet es 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"
```

Delet es the U-VNE driver configured for the device with sysOID .1.2.3.4 and running software version 12.0(23)S3 from device family 70xx in the product scheme.

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 18-12  Options and Arguments—vcb uvne delete

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysoid sysoid</td>
<td>The sysObject ID of the U-VNE configuration that you want to delete.</td>
</tr>
<tr>
<td></td>
<td><strong>Note</strong> Deleting the U-VNE does not delete or otherwise affect the U-VNE template from which the U-VNE was created.</td>
</tr>
</tbody>
</table>

Note
For the list of global options, see Global Command Options, page 18-21.

Error Codes

Table 18-13  Error Codes—vcb uvne delete

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>112</td>
<td>The sysOID does not exist or already exists and is already modeled as a VNE.</td>
</tr>
</tbody>
</table>

Note
For the list of general VCB error codes, see General Error Codes, page 18-22.

Commands for Adding and Managing Device Types

Use the following vcb devicetype commands to create, view, modify, and delete user-experience attributes for the specified device type:

- vcb devicetype add, page 18-31
- vcb devicetype view, page 18-33
- vcb devicetype modify, page 18-34
- vcb devicetype delete, page 18-35

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.

Synopsis

vcb devicetype add -devicetype device type -category prime network device category -name device name [-key device type key]-user root -password admin
Chapter 18  Enabling Support for Additional Device Types and Software Versions

Using the CLI to Create and Manage U-VNEs

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 18-23.

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 18-14 Options and Arguments—vcb devicetype add**

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devicetype device type</td>
<td>The name to assign to the new device type. Each name must be unique. To see existing device types, use the <code>vcb devicetype view</code> command.</td>
</tr>
<tr>
<td>category prime network device category</td>
<td>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.</td>
</tr>
<tr>
<td>name device name</td>
<td>The name to display for this device type in Prime Network. By default, the name is empty. <strong>Note</strong> The name need not be unique. The VCB does not enforce a naming convention for this value.</td>
</tr>
<tr>
<td>key device type key</td>
<td>(Optional) The unique ID of the new device type. This value is not displayed in Prime Network. <strong>Note</strong> We recommend that you not use this option, and let the VCB define the key instead.</td>
</tr>
</tbody>
</table>

**Note**
For the list of global options, see Global Command Options, page 18-21.
Chapter 18      Enabling Support for Additional Device Types and Software Versions

Using the CLI to Create and Manage U-VNEs

Error Codes

Table 18-15  Error Codes—vcb devicetype add

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>501</td>
<td>Device type name already exists in the deviceTypes</td>
</tr>
<tr>
<td>502</td>
<td>Category value does not match any of the possible values</td>
</tr>
<tr>
<td>503</td>
<td>Key for this template is not unique</td>
</tr>
<tr>
<td>504</td>
<td>Device type name contains illegal characters</td>
</tr>
<tr>
<td>505</td>
<td>SysOID not found or not bound to device type in devicetypes.xml</td>
</tr>
</tbody>
</table>

Note
For the list of general VCB error codes, see General Error Codes, page 18-22.

vcb devicetype view

The vcb devicetype view command returns an existing device-type association.

Synopsis

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 18-16   Options and Arguments—vcb devicetype view

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devicetype</td>
<td>device type</td>
<td>The name of the device type user-experience attributes (including name and device category) that you want to view.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tip</strong> Enter all as the devicetype to view a list of all device types defined in the system.</td>
</tr>
<tr>
<td>category</td>
<td>all</td>
<td>Returns all defined device categories (router, switch, and so on) and their numeric equivalents.</td>
</tr>
</tbody>
</table>

Note For the list of global options, see Global Command Options, page 18-21.

Error Codes

Table 18-17   Error Codes—vcb devicetype view

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>511</td>
<td>deviceType not found</td>
</tr>
</tbody>
</table>

Note For the list of general VCB error codes, see General Error Codes, page 18-22.

vcb devicetype modify

The **vcb devicetype modify** command modifies the user-experience attributes associated with specific device types.

Synopsis

```
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 18-18  Options and Arguments—vcb devicetype modify

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>devicetype</td>
<td>The name of the device type user-experience attributes that you want to modify.</td>
</tr>
<tr>
<td>category</td>
<td>(Optional) Modifies the category assigned to the device type. Enter the category as a string (router, switch, unknown, and so on).</td>
</tr>
<tr>
<td>name</td>
<td>(Optional) Modifies the name that is displayed for this device type in Prime Network.</td>
</tr>
<tr>
<td>key</td>
<td>(Optional) Modifies the unique ID of the device type. This value is not displayed in Prime Network.</td>
</tr>
</tbody>
</table>

**Note** We recommend that you *not* use this option, and let the VCB define the key instead.

Error Codes

Table 18-19  Error Codes—vcb devicetype modify

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>502</td>
<td>Category value does not match any of the possible values for this enum</td>
</tr>
<tr>
<td>503</td>
<td>Key for this device type is not unique</td>
</tr>
<tr>
<td>504</td>
<td>Device type name contains illegal characters</td>
</tr>
<tr>
<td>511</td>
<td>Device type not found</td>
</tr>
</tbody>
</table>

**Note** For the list of general VCB error codes, see General Error Codes, page 18-22.

vcb devicetype delete

The `vcb devicetype delete` command deletes the user-experience attributes that are defined for the specified device type.

**Synopsis**

```
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>
<thead>
<tr>
<th>Table 18-20</th>
<th>Options and Arguments—vcb devicetype delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Option</td>
<td>Argument</td>
</tr>
<tr>
<td>devicetype</td>
<td>device type</td>
</tr>
<tr>
<td></td>
<td>The name of the device type from which you want to delete user-experience attributes.</td>
</tr>
</tbody>
</table>

Note

For the list of global options, see Global Command Options, page 18-21.

Error Codes

<table>
<thead>
<tr>
<th>Table 18-21</th>
<th>Error Codes—vcb devicetype delete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>511</td>
<td>Device type not found.</td>
</tr>
</tbody>
</table>

Note

For the list of general VCB error codes, see General Error Codes, page 18-22.

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.

Depending on the approach that you selected to create a U-VNE, by cloning or by referencing a template, obtain technology and application support information for the cloned-from device (see the Cisco Prime Network 3.8 Administrator Guide) or the template (see U-VNE Templates, page 22-1). Use the references to do the following:

- Rigorously assess compatibility issues.
- Identify potential side effects on your device, network, and management application (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 18-39. 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 chapter, 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 Builder, 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 Test Environment, page 18-37
- Procedures for Testing and Certifying U-VNEs, page 18-39
- Troubleshooting Device Type Customizations, page 18-43

## Setting Up the 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 18-22.

### Table 18-22  Actions to Perform Before You Begin Testing

<table>
<thead>
<tr>
<th>Action</th>
<th>Purpose</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configure the device to send events to the</td>
<td>This is required before you test whether Prime Network can receive</td>
<td>Testing and Certifying Event</td>
</tr>
<tr>
<td>Prime Network server.</td>
<td>events from the VNE.</td>
<td>Customizations, page 20-12</td>
</tr>
<tr>
<td>Create an AVM for hosting the VNE instance</td>
<td>By isolating the test VNE on its own AVM, you prevent any actions that</td>
<td>Creating a Test AVM and VNE, page 18-38</td>
</tr>
<tr>
<td>on which you will perform the tests.</td>
<td>might impact the functionality of your network.</td>
<td></td>
</tr>
<tr>
<td>Measure device performance (such as</td>
<td>This enables you to determine whether the VNE is placing an</td>
<td>Measuring Device Performance, page 18-39</td>
</tr>
<tr>
<td>CPU and memory usage) on the device,</td>
<td>unreasonable load on the device.</td>
<td></td>
</tr>
<tr>
<td>with and without the VNE.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

- Setting Up the Test Environment, page 18-37
- Procedures for Testing and Certifying U-VNEs, page 18-39
- Troubleshooting Device Type Customizations, page 18-43
Chapter 18  Enabling Support for Additional Device Types and Software Versions

Testing and Certifying U-VNEs

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 “Creating an AVM” and “VNE Administration: VNE Lifecycle and Creating VNEs” in the Cisco Prime Network 3.8 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:

1. Log into Prime Network Vision.
2. Choose Network Inventory > Network Elements.
3. Perform a search to locate the device.
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 18-43.

---

**Table 18-22  Actions to Perform Before You Begin Testing (continued)**

<table>
<thead>
<tr>
<th>Action</th>
<th>Purpose</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open the relevant VNE documentation.</td>
<td>The information contained in the VNE documentation helps you during the testing process to verify how well Prime Network handles the VNE customizations you created.</td>
<td>• U-VNE Templates, page 22-1&lt;br&gt;• Cisco Prime Network 3.8 Reference Guide</td>
</tr>
<tr>
<td>Prepare a compliance report</td>
<td>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 22-1.</td>
<td>Preparing Compliance Reports, page 18-39</td>
</tr>
</tbody>
</table>
Step 5  Check the Investigation State column to see how successfully the VNE has modeled the device it represents:

- If the status is Normal, 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 18-44.

Note  For more information, see “Troubleshooting VNE Modeling” in the Cisco Prime Network 3.8 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 18-42.

Procedures for Testing and Certifying U-VNEs

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

- Preparing Compliance Reports, page 18-39
- Testing the Physical Inventory, page 18-40
- Testing the Logical Inventory, page 18-41
- Testing CPU and Memory Usage, page 18-42
- Testing That Prime Network Receives Events from an NE and Generates Tickets, page 18-42

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 3.8 Reference Guide (for a U-VNE that was cloned from an existing device family) or U-VNE Templates, page 22-1 (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).
Testing and Certifying U-VNEs

Step 9 Record your results in a compliance report. See Preparing Compliance Reports, page 18-39.

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.)

Note To delete a U-VNE, see Deleting a VNE Driver, page 18-12.

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 22-1.
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 18-39.
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:
Testing CPU and Memory Usage

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 18-39). 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:
  https://<gateway IP address>:1311
  Username: Admin
  Password: The password used to access the Prime Network applications (Manage, Vision, Events).

Testing 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.

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 using the list of supported events (listed in *Cisco Prime Network 3.8 Reference Guide* for the cloned-from device family or listed in Chapter 22, “U-VNE Templates” for a template-based U-VNE), 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 *Troubleshooting Event Customization (CLI)*, page 20-34.

---

**Troubleshooting Device Type 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 18-43
- Investigation State Not Operational, page 18-44

**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 3.8 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 18-23  Troubleshooting the Investigation State

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
<th>Action</th>
</tr>
</thead>
</table>
| 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 18-44.  
  **Note** For more information, see Prime Network Logs in *Cisco Prime Network 3.8 Administrator Guide*. |
| 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. See Deleting a VNE Driver, page 18-12.  

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 Enabling Support for a New Standard Module, page 19-1), or manage the device in Prime Network without managing that particular module.

### Moving a U-VNE to Production

**Tip** Always perform customization in a maintenance window.

Export the configuration from your lab setup and import it into the production setup using the following procedures:

- Exporting VCB Registry Customizations, page 21-3
- **Importing VCB Registry Customizations, page 21-3**

After you import the changes to the production server:

1. Restart Prime Network or at a minimum, restart all VNEs and AVMs that need to support the customizations.

2. Repeat testing and certification to ensure that the customizations are functioning as expected in your production environment. See **Procedures for Testing and Certifying U-VNEs, page 18-39**.
Enabling Support for Additional Modules

You can use the Prime Network VCB to enable support new standard and pluggable modules on developed VNEs and U-VNEs.

Topics include:

- Enabling Support for a New Standard Module, page 19-1
- Viewing and Managing Existing Standard Modules, page 19-3
- Enabling Support for a New Pluggable Module, page 19-5
- Managing Existing Pluggable Modules, page 19-6
- Testing and Certifying Module Customization, page 19-9
- Using the CLI to Add and Manage Modules, page 19-9

Enabling 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.

The following table outlines the workflow for enabling support for a new standard module.
Table 19-1  Steps for Adding Support for a Standard Module

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td>Research the capabilities of the module. Search the website of the device manufacturer, or use standard search tools, such as Google. For example, to access data sheets for the cards and modules supported by Cisco 12000 Series Routers, use this URL: <a href="http://www.cisco.com/en/US/products/hw/routers/ps167/products_relevant_interfaces_and_modules.html">http://www.cisco.com/en/US/products/hw/routers/ps167/products_relevant_interfaces_and_modules.html</a></td>
</tr>
</tbody>
</table>
| **Step 2** | Obtain the module identifier. Do one of the following:  
• Examine the containing device in Prime Network Vision.  
• Look in Cisco MIB support documentation.  
• Connect to the device and issue an SNMP query to obtain the OID.  
Note: The OID is dependent on the MIB used by the template. For example, for Cisco devices that use the Entity MIB, use .1.3.6.1.2.1.47.1.1.1.1.3. |
| **Step 3** | Use the VCB to add the module. Use the VCB to create a module definition that binds the template to the module identifier. This definition is added to the module group file that contains the template. See Adding Modules - Procedure, page 19-2. |
| **Step 4** | Restart the AVM that manages the developed VNE or U-VNE with the module. Restart the AVM that manages the relevant VNE. Prime Network can then begin modeling the new module. See the Cisco Prime Network 3.8 Administrator Guide. |
| **Step 5** | Test the developed VNE or U-VNE. Test the VNE to ensure that the module is correctly represented in Prime Network. If it is not, you must delete the module definition and try again using a different module template. See Testing and Certifying Module Customization, page 19-9 |

Adding Modules - Procedure

To add support for a new module:

**Step 1**  
In the VCB tool, select **Tools > Modules.**  
The Modules tab displays the modules under the default module group, cisco-catalyst-spec.

**Step 2**  
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 3**  
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:
Viewing and Managing Existing Standard Modules

The Prime Network VCB GUI enables you to manage existing standard modules. You can perform the following actions to manage the modules:

- Viewing Modules, page 19-3
- Editing Standard Modules, page 19-4
- Deleting Standard Modules, page 19-5

Viewing Modules

To view a list of modules:

**Step 1** In the VCB tool, select **Tools > Modules**.

Modules are displayed per module group, as shown in Figure 19-1.

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

**Step 2** Click on the arrow next to a module to expand its display and see all the port layers supported by that module.
Chapter 19  Enabling Support for Additional Modules

Viewing and Managing Existing Standard Modules

Figure 19-1  Viewing Modules in Prime Network VCB

You can perform the following tasks using the Modules Toolbar:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Task</th>
<th>Related Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Edit Icon]</td>
<td>Edit</td>
<td>Editing Standard Modules, page 19-4</td>
</tr>
<tr>
<td>![Delete Icon]</td>
<td>Delete a user defined module or restore a system default module</td>
<td>Deleting Standard Modules, page 19-5</td>
</tr>
<tr>
<td>![Add Row Icon]</td>
<td>Add Row</td>
<td>Adding Modules - Procedure, page 19-2</td>
</tr>
<tr>
<td>![Filter Icon]</td>
<td>Filter</td>
<td>Filtering the VCB Tables, page 17-3</td>
</tr>
</tbody>
</table>

Related Topic

Viewing Pluggable Modules, page 19-7

Editing Standard Modules

Both user-defined and system default modules can be edited. 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.

To edit a module:

**Step 1**  In the VCB tool, select **Tools > Modules**.
Step 2 Select the module you want to edit, by doing one of the following:
- Double-click on the row
- Select the radio button on the left of the row and then click the Edit button
The selected row appears in editable mode.

Step 3 Edit the module as required.

Step 4 Click Save.

Related Topic
Editing Pluggable Modules, page 19-7

Deleting Standard Modules

Only user-defined modules can be deleted.
To delete a module:

Step 5 In the VCB tool, select Tools > Modules.

Step 6 Select the module you want to delete.

Step 7 Click the Delete button on the toolbar.

Note If a system default module has been edited, a Restore button will appear in place of the Delete button, enabling you to revert back to the values of the system default module.

Step 8 In the confirmation dialog box that appears, click OK to delete the module. Otherwise click Cancel.

Related Topic
Deleting Pluggable Modules, page 19-8

Enabling Support for a New Pluggable Module

To add a new pluggable module to the Prime Network VCB:

Step 1 Log into the Prime Network VCB GUI using your login credentials.
On successful login, Prime Network VCB GUI displays the home page.

Step 2 Choose Tools > Pluggable Modules.
The Pluggable Modules tab displays all the pluggable modules in Prime Network VCB.

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:
**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:

---

**Managing Existing Pluggable Modules**

Using the Prime Network VCB GUI you can add, edit, and delete modules. The step-by-step procedure for these tasks are given under the topics:

- Viewing Pluggable Modules, page 19-7
- Editing Pluggable Modules, page 19-7
- Deleting Pluggable Modules, page 19-8

Prime Network VCB allows you to manage pluggable modules using the CLI and the GUI. Table 19-2 lists the references to the specific tasks you can perform:
Chapter 19 Enabling Support for Additional Modules

Managing Existing Pluggable Modules

Table 19-2 References to Tasks in Prime Network VCB

<table>
<thead>
<tr>
<th>Tasks</th>
<th>GUI Support</th>
<th>CLI Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Viewing Pluggable Modules</td>
<td>Viewing Modules, page 19-3</td>
<td>Viewing a Pluggable Module</td>
</tr>
<tr>
<td>Adding Pluggable Modules</td>
<td>Enabling Support for a New Standard Module, page 19-1</td>
<td>Managing an Additional Pluggable Module</td>
</tr>
<tr>
<td>Editing Pluggable Modules</td>
<td>Editing Pluggable Modules, page 19-7</td>
<td>VCB PluggableModule Modify (Command Reference Only)</td>
</tr>
<tr>
<td>Deleting Pluggable Modules</td>
<td>Deleting Pluggable Modules, page 19-8</td>
<td>VCB PluggableModule Delete (Command Reference Only)</td>
</tr>
</tbody>
</table>

Viewing Pluggable Modules

To view the pluggable modules in the Prime Network VCB GUI:

**Step 1** Log into the Prime Network VCB GUI using your login credentials.

On successful login, Prime Network VCB GUI displays the home page.

**Step 2** Select **Tools > Pluggable Modules**.

The pluggable modules are displayed in Prime Network VCB.

You can perform the following tasks using the Pluggable Modules toolbar:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Task</th>
<th>Related Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="icon" alt="Edit" /></td>
<td>Edit</td>
<td>Editing Pluggable Modules, page 19-7</td>
</tr>
<tr>
<td><img src="icon" alt="Delete" /></td>
<td>Delete</td>
<td>Deleting Pluggable Modules, page 19-8</td>
</tr>
<tr>
<td><img src="icon" alt="Add Row" /></td>
<td>Add Row</td>
<td>Enabling Support for a New Pluggable Module, page 19-5</td>
</tr>
<tr>
<td><img src="icon" alt="Filter" /></td>
<td>Filter</td>
<td>Filtering the VCB Tables, page 17-3</td>
</tr>
</tbody>
</table>

**Related Topic**

Viewing Modules, page 19-3

Editing Pluggable Modules

To edit the pluggable module properties:

**Step 1** Log into the Prime Network VCB GUI using your login credentials.
On successful login, Prime Network VCB GUI displays the home page.

**Step 2**  Choose **Tools > Pluggable Modules**.

**Step 3**  Select the module which you want to edit, by doing one of the following:
- Click on the row
- Select the radio button to the left of the row and then click the **Edit** button

The selected row appears in editable mode.

**Step 4**  To change the module template, click on the arrow on the template field.

**Step 5**  Choose the required template from the drop-down list that appears.

**Step 6**  Click **Save**.

**Step 7**  To view the edited changes in the module:
- Restart the AVM, that manages the devices with pluggable ports.
- Test the VNEs. For the procedure on testing the VNEs, see **Testing and Certifying Module Customization**, page 19-9.
- Restart the Prime Network server.

**Note**  You cannot edit Prime Network default pluggable modules.

**Related Topic**
**Editing Standard Modules**, page 19-4

### Deleting Pluggable Modules

To delete a pluggable module:

**Step 1**  Log into the Prime Network VCB GUI using your login credentials.

On successful login, Prime Network VCB GUI displays the home page.

**Step 2**  Choose **Tools > Pluggable Modules**.

**Step 3**  To select the module which you want to delete, select the radio button on the left of that row.

**Step 4**  Click on the **Delete** button on the taskbar.

**Step 5**  In the confirmation dialog box that appears, click **OK** to delete the module. Otherwise click **Cancel**.

**Step 6**  Restart the AVM, that manages the devices with pluggable ports.

**Step 7**  Test the VNEs. For the procedure on testing the VNEs, see **Testing and Certifying Module Customization**.

**Step 8**  Restart the Prime Network server.

**Note**  You cannot delete Prime Network VCB default pluggable modules.
Testing and Certifying Module Customization

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 Test Environment, page 18-37. 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 Chapter 21, “Deploying and Maintaining Extensions in a Prime Network Production System”.

Using the CLI to Add and Manage Modules

You have the option to use the CLI to perform VCB customizations. Use the CLI if you are an advanced VCB user or if the current VCB GUI does not support the customization you need to perform.

For information about general VCB commands, see Introductory Command Reference and Global Options, page 18-20.

This section describes the CLI commands that can be used to add and manage modules, as follows:
  • Planning—Evaluating Module Templates, page 19-9
  • Commands for Managing Modules, page 19-10
  • Commands for Managing Pluggable Modules, page 19-18

Planning—Evaluating Module Templates

When adding support for a new module using the VCB, you must identify the module template that matches the capabilities of the module. Each module template consists of specific definitions for each port layer.

For example, the atm-default module template contains the following port layer definitions:
  • 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.

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 22-10.

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 22-14 to identify the module template that best matches the module. You can then add support for the module using the VCB. See Enabling Support for a New Standard Module, page 19-1.

Commands for Managing Modules

Use the following vcb module commands to add, view, modify, and delete modules from VNEs:

- vcb module add, page 19-11
- vcb module view, page 19-13
- vcb module modify, page 19-15
- vcb module delete, page 19-16

Note

To create, view, modify, and delete pluggable modules, see Commands for Managing Pluggable Modules, page 19-18.
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.

**Synopsis**

```
vcb module add -module module identifier -template template name -group module group 
[ -hardwareDescription 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> [ -hardwareDescription 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 -hardwareDescription cevCat6kWsf6kpf3b -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 cevCat6kWsf6kpf3b.

**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 19-3  Options and Arguments—vcb module add

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>module identifier</td>
<td>The module identifier, a number, string, or OID. The module identifier should be unique and new.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> The value specified for this argument must match the value returned from the physical command investigation.</td>
</tr>
<tr>
<td>group</td>
<td>module group</td>
<td>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 22-10.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> Regular modules and pluggable modules use different module spec files. For information about pluggable modules, see Commands for Managing Pluggable Modules, page 19-18.</td>
</tr>
<tr>
<td>sysoid</td>
<td>sysoid</td>
<td>The sysObject ID of the device that belongs to particular device family for which a concrete module is created using the implementation defined in the template.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> This sysObject ID must already exist in the system.</td>
</tr>
<tr>
<td>scheme</td>
<td>scheme</td>
<td>Scheme name which will be used while adding the VNE.</td>
</tr>
<tr>
<td>template</td>
<td>template name</td>
<td>The name of the template that contains the implementation of the physical investigation of this module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tip</strong> The template option need not be specified if the module does not support any ports; SIP and other carrier cards are examples of modules that do not support ports.</td>
</tr>
<tr>
<td>hardwareDescription</td>
<td>hardware description</td>
<td>(Optional) Hardware description of the given module.</td>
</tr>
</tbody>
</table>

Note

For the list of global options, see Global Command Options, page 18-21.

Error Codes

Table 19-4  Error Codes—vcb module add

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>No such template name in the templates file.</td>
</tr>
<tr>
<td>111</td>
<td>The SysOID specified does not exist in site.xml.</td>
</tr>
<tr>
<td>401</td>
<td>Module already exists in the spec file or in site.xml (thus is already modeled).</td>
</tr>
<tr>
<td>402</td>
<td>Module template file not found.</td>
</tr>
</tbody>
</table>
vcb module view

The `vcb module view` command returns an existing module configuration.

**Synopsis**

```
vcb module view -sysoid sysoid -module {module identifier | all} -user root -password admin
-vscheme <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.

---

**Note**

For the list of general VCB error codes, see `General Error Codes, page 18-22`. 

---
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>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysoid</td>
<td><code>sysoid</code></td>
<td>The sysOID of the device that contains the module configuration that you want to view.</td>
</tr>
<tr>
<td>module</td>
<td><code>module identifier</code></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tip</strong> Enter <code>all</code> as the <code>module identifier</code> to return a list of all modules in the defined device or group, listed by identifier.</td>
</tr>
<tr>
<td>group</td>
<td><code>module group</code></td>
<td>The module spec group associated with the module or template whose details you want to view.</td>
</tr>
<tr>
<td>template</td>
<td><code>template name</code></td>
<td>The name of the module template whose port layer configuration you want to view.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Tip</strong> Enter <code>all</code> as the <code>template name</code> to return the list of all module templates contained in the specified module spec group.</td>
</tr>
<tr>
<td></td>
<td><code>- scheme scheme</code></td>
<td>Scheme name which will be used while adding the VNE.</td>
</tr>
</tbody>
</table>

For the list of global options, see Global Command Options, page 18-21.

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>No such template name in the template file</td>
</tr>
<tr>
<td>402</td>
<td>Module template file not found</td>
</tr>
<tr>
<td>411</td>
<td>SysOID specified does not exist in site.xml or vendor file</td>
</tr>
<tr>
<td>412</td>
<td>Module identifier specified does not exist</td>
</tr>
</tbody>
</table>

For the list of general VCB error codes, see General Error Codes, page 18-22.
vcb module modify

Use the vcb module modify command to modify the associated module template.

Synopsis

```
vcb module modify -module module identifier -group module group -template template name
   [ -hardwareDescription Hardware Description of the Module -user username -password password]

vcb module modify -module module identifier -sysoid device sysoid -scheme <ipcore|product> -template template name [ -hardwareDescription 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 cc3-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>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>module identifier</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> The value specified for this argument must match the value returned by the device when investigating this module.</td>
</tr>
<tr>
<td>group</td>
<td>module group</td>
<td>The name of the group that contains the module template to apply to the module.</td>
</tr>
</tbody>
</table>
Chapter 19  Enabling Support for Additional Modules

Using the CLI to Add and Manage Modules

Table 19-7  Options and Arguments—vcb module modify (continued)

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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. |
| hardwareDescription 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 18-21.

Error Codes

Table 19-8  Error Codes—vcb module modify

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>No such template name in the template file</td>
</tr>
<tr>
<td>402</td>
<td>Module template file not found</td>
</tr>
<tr>
<td>411</td>
<td>SysOID specified does not exist in site.xml or vendor file</td>
</tr>
<tr>
<td>412</td>
<td>Module identifier specified not found or already exists in vendor spec file</td>
</tr>
</tbody>
</table>

**Note** For the list of general VCB error codes, see General Error Codes, page 18-22.

vcb module delete

Use the vcb module delete command to delete the association between a module and the implementation defined in a module template.

**Synopsis**

```
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

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 19-9**  Options and Arguments—vcb module delete

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module module identifier</td>
<td>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.</td>
</tr>
<tr>
<td>sysoid device sysoid</td>
<td>The sysObject ID of the device that contains the module configuration that you want to delete.</td>
</tr>
<tr>
<td>group module group</td>
<td>The module spec group that contains the template whose implementation you want to delete from the module.</td>
</tr>
<tr>
<td>scheme scheme</td>
<td>Scheme name which will be used while adding the VNE.</td>
</tr>
</tbody>
</table>

**Note**

For the list of global options, see [Global Command Options, page 18-21](#).

Error Codes

**Table 19-10**  Error Codes—vcb module delete

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>402</td>
<td>Module template file not found.</td>
</tr>
<tr>
<td>411</td>
<td>sysOID does not exist in the site.xml file or vendor spec file.</td>
</tr>
<tr>
<td>412</td>
<td>Module identifier not found or already exists in the vendor spec file.</td>
</tr>
</tbody>
</table>

**Note**

For the list of general VCB error codes, see [General Error Codes, page 18-22](#).

Commands for Managing Pluggable Modules

Use the following `vcb pluggablemodule` commands to add, view, modify, and delete pluggable modules:

- `vcb pluggablemodule add`, page 19-19
- `vcb pluggablemodule view`, page 19-20
- `vcb pluggablemodule modify`, page 19-22
- `vcb pluggablemodule delete`, page 19-23

**Note**

To add, view, modify, or delete regular modules, see [Commands for Managing Modules, page 19-10](#).
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.

**Synopsis**

```
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 19-11 Options and Arguments — vcb pluggablemodule add

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>moduleno</td>
<td>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. <strong>Note</strong> The value specified for this argument must match the value returned by the device when investigating this module.</td>
</tr>
<tr>
<td>group</td>
<td>pluggable port group</td>
<td>The name of the group.</td>
</tr>
<tr>
<td>pid</td>
<td>pid</td>
<td>The pid for the module.</td>
</tr>
<tr>
<td>mediatype</td>
<td>MediaType</td>
<td>The media type of the port on the pluggable module.</td>
</tr>
<tr>
<td>pluggable type</td>
<td>pluggableType</td>
<td>The type of the pluggable module; one of SFP, XFP, X2, XENPAK.</td>
</tr>
<tr>
<td>containeroid</td>
<td></td>
<td>The oid of a pluggable module container where the pluggable module will be inserted.</td>
</tr>
<tr>
<td>basetype</td>
<td></td>
<td>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</td>
</tr>
</tbody>
</table>

**Note** For the list of global options, see Global Command Options, page 18-21.

### Error Codes

#### Table 19-12 Error Codes — vcb pluggablemodule add

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>403</td>
<td>Pluggable module spec file not found</td>
</tr>
<tr>
<td>404</td>
<td>Invalid pluggable module type</td>
</tr>
</tbody>
</table>

**Note** For the list of general VCB error codes, see General Error Codes, page 18-22.

---

vcb pluggablemodule view

Use the **vcb pluggablemodule view** command to display details for a pluggable module.

### Synopsis

```
vcb pluggablemodule view -group groupName -module moduleno | all [-userdefined] -user username -password password
```
**vcb pluggablemodule view**

```
 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

vcb pluggablemodule view -group pluggable-port-spec -containeroid all -user root
-password admin

vcb pluggablemodule view -group pluggable-port-spec -basetype all -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.

Returns the pluggable module container oid details, if the list has userdefined container oid, it is marked with # after the oid.

Returns the pluggable module details.

**Options**

**Table 19-13 Options and Arguments—vcb pluggablemodule view**

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module moduleno</td>
<td>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.</td>
</tr>
<tr>
<td>group groupname</td>
<td>The name of the group.</td>
</tr>
</tbody>
</table>

**Note**

For the list of global options, see **Global Command Options, page 18-21.**

**Error Codes**

**Table 19-14 Error Codes—vcb pluggablemodule view**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>403</td>
<td>Pluggable module spec file not found.</td>
</tr>
</tbody>
</table>
vcb pluggablemodule modify

Use the **vcb pluggablemodule modify** command to change the pluggable module configuration.

**Synopsis**

```
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 19-15 Options and Arguments—vcb pluggablemodule modify**

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module <em>moduleno</em></td>
<td>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.</td>
</tr>
<tr>
<td>group <em>pluggable port group</em></td>
<td>The name of the group.</td>
</tr>
<tr>
<td>pid <em>pid</em></td>
<td>The pid for the module.</td>
</tr>
<tr>
<td>mediatype <em>MediaType</em></td>
<td>(Optional) The media type of the port on the pluggable module.</td>
</tr>
<tr>
<td>pluggable type <em>pluggableType</em></td>
<td>(Optional) The type of the pluggable module; one of SFP, XFP, X2, AND XENPAK.</td>
</tr>
</tbody>
</table>

**Note**

For the list of general VCB error codes, see [General Error Codes, page 18-22](#).
**vcb pluggablemodule delete**

Use the `vcb pluggablemodule delete` command to delete a pluggable module that was previously added using the VCB.

**Synopsis**

```
vcb pluggablemodule delete -group pluggableportgroupN -module modulenumber -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>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>module</td>
<td>moduleno</td>
<td>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.</td>
</tr>
<tr>
<td>group</td>
<td>pluggable port group</td>
<td>The name of the group from which to delete the pluggable module.</td>
</tr>
</tbody>
</table>

**Note**

For the list of global options, see [Global Command Options, page 18-21](#).
Managing Additional Event Types Using the VCB

This chapter contains the following sections:

- Overview, page 20-1
- Enabling Support for Additional Events, page 20-2
- Customizing Events, page 20-10
- Testing and Certifying Event Customizations, page 20-12
- Using the CLI to Customize Events, page 20-13
- Troubleshooting Event Customization (CLI), page 20-34

### Overview

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.

You can customize events using the VCB tool within Prime Network, or using VCB CLI commands. This chapter primarily describes event customization using the Prime Network VCB GUI. For information on using the CLI method, see Using the CLI to Customize Events, page 20-13.

The Events tab in the VCB tool provides all the functionality required to add and customize events.
Enabling Support for Additional Events

Using the VCB, you can enable Prime Network to recognize currently unsupported traps or syslogs. The procedures for adding support for events are illustrated through the following example use cases:

Enabling Support for Unsupported Traps, page 20-2
Adding Support for a Custom Syslog, page 20-7

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.

Use the VCB to add support for unsupported traps, 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.
Chapter 20  Managing Additional Event Types Using the VCB

Enabling Support for Additional Events

Figure 20-2  Expanded VNE Driver Properties Showing Parsing Rules

Step 3  Click the Trap Parsing Rule link to show a list of traps associated with this parsing rule.

Figure 20-3  Traps Associated with cisco-trap-product-parsing-rules

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. The file extension should be .mib or .my or no extension.

A list of MIB dependencies is displayed. A green check mark indicates that the dependency file has been found on the server.
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 unsupported trap for which you want to add support, and click Support.
Figure 20-5  List of Supported and Unsupported Traps

An event is created and the first page of the Event Definition wizard is displayed. The event name and OID are pre-populated but can be changed if necessary. The OID is the common prefix of the OIDs of the selected subtypes.

**Step 9**  Click Next to go to the next step in the wizard which is to define the event subtypes. You will see that a subtype has been created for each unsupported trap you selected. Edit the information for each subtype as required:

- **Name**—Enter a unique name for the event subtype.
- **Description**—Enter a string that describes the event subtype.
- **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.

**Note**  If you make a sub-type ticketable, a ticket will be generated for it. When a non-ticketable sub-type 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.
Step 10  Click Next to go to the next step in the wizard which is subtype identification. In this step, you define how Prime Network will differentiate between the subtypes, as follows:

- By Trap OID—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 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 11  Click Next to go to the next step in the wizard which is association of the event with the VNE. Specify the following information:

- Source Type—Select 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.
  - Efp Key From Ifname Serviceid—Associates the event with a specific EFP DC, based on the service instance ID and the interface name.
  - 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.
- 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 12  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 13 Click **Add** to select additional parsing rule groups, as required. 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 20-10 for the relationship between parsing rule groups.

**Note** For changes to take effect, you must restart the VNEs that are affected.

### Adding 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** Click **Next** to go to the next step in the wizard which is to define the event subtypes.
Step 7  Enter the following information to define the first event subtype:

- **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 8  Click **Add** to define a second subtype, for example, XP server accessible, with severity “Cleared”.

**Figure 20-7  Subtype Definition**

![Subtype Definition](image)

Step 9  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 10 Enter the following information to define event identification and association:

- **Raw Event**—Provide the raw event syntax as an example, so that the system can parse it.
- **Subtype Key**—Keyword that identifies the subtype. The keyword should be taken from the raw event. In this example, the key would be “unreachable”.

---

* Cisco Prime Network 3.8 Customization Guide

OL-24647-01
• Source Type—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:
  – Efp Key From Ifname Serviceid—Associates the event with a specific EFP DC, based on the service instance ID and the interface name.
  – 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 With SlotSubslot Value Key—Associates the event with the corresponding module, based on the slot number.

• 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.

• Interface Identifier—Specify a value by which the interface will be identified (ifIndex is used to identify the interface).

• Interface Name—Specify the name of the interface to which the event is associated ((ifName is used to identify the interface).

• Instance Identifier—Specify the identifier of the instance. For this example, the instance identifier could be Windows XP.

Step 11 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 12 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 13 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 20-10 for the relationship between parsing rule groups.

Step 14 Click Finish. The event now appears in the list of syslogs for the cisco-syslog-product-parsing-rules group.
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 20-10
- Dropping an Event, page 20-11
- Restoring a System Default Event, page 20-11
- Deleting Events, page 20-12

### 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.

### Parsing Rule Group Inheritance Structure

<table>
<thead>
<tr>
<th>Parsing Rule Group</th>
<th>Inherits From...</th>
</tr>
</thead>
<tbody>
<tr>
<td>cisco-asr90xx-syslog-ipcore-parsing-rules</td>
<td>cisco-iox-syslog-ipcore-parsing-rules¹</td>
</tr>
<tr>
<td>cisco-asr90xx-trap-ipcore-parsing-rules</td>
<td>cisco-iox-trap-ipcore-parsing-rules</td>
</tr>
<tr>
<td>cisco-ciscopt-trap-ipcore-parsing-rules</td>
<td>cisco-trap-ipcore-parsing-rules</td>
</tr>
<tr>
<td>cisco-iox-syslog-ipcore-parsing-rules</td>
<td>cisco-iox-syslog-product-parsing-rules</td>
</tr>
<tr>
<td>cisco-syslog-ipcore-parsing-rules</td>
<td>cisco-syslog-product-parsing-rules</td>
</tr>
<tr>
<td>cisco-trap-ipcore-parsing-rules</td>
<td>cisco-trap-product-parsing-rules</td>
</tr>
</tbody>
</table>

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.

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 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 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:

1. In the VCB tool, select **Tools > Events** or click on the Events tab.
2. Select the event you want to delete and click **Delete**.
3. Click **OK** in the confirmation message.

The event is removed from the table.

### Testing and Certifying Event Customizations

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

To prepare your test environment:

1. Restart Prime Network or restart at least all VNEs/AVMs that need to support the new event. A restart is required because event patterns are loaded upon VNE initialization only.
2. If using a simulator, enable Prime Network to process traps sent from a simulator by running this command and restarting avm100:

   ```bash
   runRegTool.sh -gs localhost set 0.0.0.0 avm100/agents/trap/processors/snmp-processor/class com.sheer.metrocentral.framework.instrumentation.trap.processor.RawAgentIpSnmpEventProcessor
   ```

3. 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:

   ```bash
   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 **Moving a Tested and Certified Event Customization to Production**, page 20-13.

### Moving a Tested and Certified Event Customization to Production

**Tip** Always perform customization during a scheduled maintenance window.

Export the configuration from your lab setup and import it into the production setup using the following procedures:

- Exporting VCB Registry Customizations, page 21-3
- Importing VCB Registry Customizations, page 21-3

After you import the changes to the production server:

1. Restart Prime Network or at a minimum, restart all VNEs and AVMs that need to support the new events.
2. Repeat testing and certification to ensure that the customizations are functioning as expected in your production environment. See **Testing and Certifying Event Customizations**, page 20-12

### Using the CLI to Customize Events

You have the option to use the CLI to perform VCB customizations. Use the CLI if you are an advanced VCB user or if the current VCB GUI does not support the customization you need to perform.

For information about general VCB commands, see **Introductory Command Reference and Global Options**, page 18-20.

This section describes the CLI commands that can be used to customize events, as follows:

- `vcb event Commands`, page 20-14
- `vcb eventparsingrules Commands`, page 20-21
- `vcb eventpattern Commands`, page 20-28
- `vcb eventarg Commands`, page 20-33
vcb event Commands

Use the following vcb event commands to create, view, modify, and delete events:

- vcb event add, page 20-14
- vcb event view, page 20-16
- vcb event modify, page 20-18
- vcb event delete, page 20-20

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 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 20-16.

Synopsis

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]
...
{-subtype n subtypenName} [-ticketable n][-severity n critical|major|minor|warning|info|cleared]
[-shortdesc n short description string] [-autoclear n 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

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>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| eventtype event type | Type of event. Valid values are:  
  - syslog  
  - trap |
| eventname event name | Unique string that identifies the event within Prime Network. |
| 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 subtype name | Unique string that identifies the subevent with the Prime Network event. |
| ticketable | (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 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. For information about root cause events, see “Fault Management” and “Causality Correlation” in Cisco Active Network Abstraction 3.7.2 Theory of Operations.  
  By default, autoclear is true for user-defined event definitions. |
| severity severity level | Mandatory parameter for subtype. The severity of the subevent.  
  Possible values are critical, major, minor, warning, info, and cleared. |
  This string is stored in the event database.  
  By default, the subtype value is used as the default shortdesc value. |

**Note** For the list of global options, see Global Command Options, page 18-21.
Chapter 20  Managing Additional Event Types Using the VCB

Using the CLI to Customize Events

Error Codes

Table 20-3  Error Codes—vcb event add

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>201</td>
<td>Event name already exists in Prime Network.</td>
</tr>
<tr>
<td>202</td>
<td>Alarm ID already exists in Prime Network.</td>
</tr>
</tbody>
</table>

Note
For the list of general VCB error codes, see General Error Codes, page 18-22.

vcb event view

Use the vcb event view command to list event definition registrations.

Synopsis

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`
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 20-4  Options and Arguments—vcb event view**

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventname</td>
<td>eventName</td>
<td>Unique string that represents the event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tip Enter <strong>all</strong> 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.</td>
</tr>
<tr>
<td>substringmatch</td>
<td></td>
<td>(Optional) Indicates that the event name argument is not an exact match.</td>
</tr>
<tr>
<td>eventtype</td>
<td>trap</td>
<td>syslog</td>
</tr>
<tr>
<td>genericevents</td>
<td>generic</td>
<td>event type</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tip Enter <strong>all</strong> as the generic event type to display information on all the events in Prime Network.</td>
</tr>
<tr>
<td>ipaddress</td>
<td>neip</td>
<td>(Optional) Generic events filter. IP address for the NE for which you want to see generic events.</td>
</tr>
<tr>
<td>date</td>
<td>yyyy-mm-dd</td>
<td>(Optional) Generic events filter. The date after which the events arrived. (Returns events that arrived after the given day.)</td>
</tr>
<tr>
<td>time</td>
<td>hh:mm:ss</td>
<td>(Optional) Generic events filter. The time after which the events arrived. (Returns events that arrived after the given time.)</td>
</tr>
<tr>
<td>maxrecords</td>
<td>num</td>
<td>(Optional) Generic events filter. The maximum number of events that you want to display. The default value is 100.</td>
</tr>
<tr>
<td>mibfile</td>
<td>complete-path-</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>mibFilename</td>
<td><strong>Note</strong> 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.</td>
</tr>
<tr>
<td>generatecli</td>
<td></td>
<td>(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 <strong>-mibfile</strong> option.</td>
</tr>
</tbody>
</table>
Using the CLI to Customize Events

Chapter 20  Managing Additional Event Types Using the VCB

Note
For the list of global options, see Global Command Options, page 18-21.

Error Codes

Table 20-5  Error Codes—vcb event view

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>231</td>
<td>No such event exists in the events file</td>
</tr>
</tbody>
</table>

Note
For the list of general VCB error codes, see General Error Codes, page 18-22.

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.

Synopsis

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...
{ -subtype1 subtype1Name } { [-ticketable1] } [-autoclear1 false|true]
- severity1n critical|major|minor|warning|info|cleared
{ -shortdesc1n } 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

```bash
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 20-14 for the `vcb event add`.

```bash
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>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>eventName</td>
<td>eventName</td>
<td>Unique string identifies the event within Prime Network.</td>
</tr>
<tr>
<td>eventtype</td>
<td>trap</td>
<td>syslog</td>
</tr>
<tr>
<td>alarmid</td>
<td>alarmId</td>
<td>(Optional) Unique integer identifier for the event. If not provided, the VCB automatically generates a unique number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> We recommend that you do not provide an input alarm ID.</td>
</tr>
<tr>
<td>subtypen</td>
<td>subtypenName</td>
<td>Unique string that identifies the subevent with the Prime Network event.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>ticketableen</td>
<td>(Optional) Parameter for subtype. Indicates whether a ticket should be generated for this subtype. If not specified, no ticket is generated.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong> 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.</td>
</tr>
</tbody>
</table>
Using the CLI to Customize Events

**Table 20-6 Options and Arguments—vcb event modify (continued)**

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-autoclear, false</td>
<td>(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.</td>
</tr>
<tr>
<td>severity value</td>
<td>(Optional) Parameter for subtype. Specifies the severity of the subevent. Possible values are critical, major, minor, warning, info, and cleared.</td>
</tr>
<tr>
<td>override</td>
<td>(Optional) Indicates that you expect to override attributes for a factory-defined event.</td>
</tr>
</tbody>
</table>

**Note**
For the list of general VCB error codes, see General Error Codes, page 18-22.

**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.

**Synopsis**

```
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>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>eventname</code></td>
<td><code>eventName</code></td>
</tr>
<tr>
<td><code>eventName</code></td>
<td>Name of the event to be deleted</td>
</tr>
</tbody>
</table>

**Note**

For the list of global options, see *Global Command Options*, page 18-21.

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>231</td>
<td>No such event exists in the events file</td>
</tr>
</tbody>
</table>

**Note**

For the list of general VCB error codes, see *General Error Codes*, page 18-22.

**vcb eventparsingrules Commands**

Use the following `vcb eventparsingrules` commands to create, view, modify, and delete event parsing rules:

- `vcb eventparsingrules add`, page 20-22
- `vcb eventparsingrules view`, page 20-24
- `vcb eventparsingrules modify`, page 20-26
- `vcb eventparsingrules delete`, page 20-27
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.

Synopsis

```bash
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

```bash
vcb eventparsingrules add
 -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_textmessage "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.
Using the CLI to Customize Events

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 uniquiid 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 20-10 Options and Arguments—vcb eventparsingrules add**

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templates</td>
<td>template nametemplates template nametemplate name1, ... template namen</td>
<td>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.</td>
</tr>
<tr>
<td>group</td>
<td>repository</td>
<td>Specifies the vendor-specific trap or syslog repository file under which the customizations should be made.</td>
</tr>
<tr>
<td>rule</td>
<td>rulename</td>
<td>String that is used as a key name for event rule definition.</td>
</tr>
<tr>
<td>enable</td>
<td></td>
<td>(Optional) Indicates whether the rule is enabled or disabled. Only enabled rules are used to parse incoming traps and syslogs.</td>
</tr>
<tr>
<td>variable arguments</td>
<td></td>
<td>Arguments vary from one event template to another event template.</td>
</tr>
<tr>
<td>syslog_identification_testmessage</td>
<td></td>
<td>(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.</td>
</tr>
</tbody>
</table>
For the list of global options, see Global Command Options, page 18-21.

Error Codes

Table 20-11  Error Codes—vcb eventparsingrules add

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>Event template file not found.</td>
</tr>
<tr>
<td>103</td>
<td>No such template name in template file.</td>
</tr>
<tr>
<td>212</td>
<td>Only one template can be selected from each template category.</td>
</tr>
<tr>
<td>213</td>
<td>Invalid expression for syslog.</td>
</tr>
</tbody>
</table>

For the list of general VCB error codes, see General Error Codes, page 18-22.

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).

Synopsis

vcb eventparsingrules view -group repository name -rulename { rulename | all } [-detail]
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>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group repository name</td>
<td>The trap or syslog repository filename.</td>
</tr>
<tr>
<td>rulename ruleName</td>
<td>The unique string that is used to represent the event parsing rules. <strong>Tip</strong> Enter all as the ruleName to display information on all the rules in Prime Network.</td>
</tr>
<tr>
<td>detail</td>
<td>(Optional) Lists the entire rule contents including the parsing rule entry details.</td>
</tr>
<tr>
<td>template templateName</td>
<td>The event template name. <strong>Tip</strong> Enter all as the templateName to display information on all event templates in Prime Network.</td>
</tr>
<tr>
<td>inputparam</td>
<td>(Optional) Lists template definition entries that require user input when creating event parsing rules.</td>
</tr>
</tbody>
</table>

### Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>No such template name in the templates file.</td>
</tr>
<tr>
<td>222</td>
<td>Parsing rules repository not found.</td>
</tr>
<tr>
<td>231</td>
<td>No such rule name in the site.xml.</td>
</tr>
</tbody>
</table>

**Note**

For the list of general VCB error codes, see General Error Codes, page 18-22.
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.

**Synopsis**

```bash
vcb eventparsingrules modify -template templateName1, templateName2, ..., templateNamen -group repository -rulename rulename [-enable] [-example syslogMessage]
{ [-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>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>templates</td>
<td>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.</td>
</tr>
<tr>
<td>templateName1, ... templateNamen</td>
<td></td>
</tr>
<tr>
<td>group repository</td>
<td>The hive under which the customizations should be made. The hive is the vendor-specific trap or syslog repository file.</td>
</tr>
<tr>
<td>rulename ruleName</td>
<td>String that is used as a key name for event rule definition.</td>
</tr>
<tr>
<td>enable</td>
<td>(Optional) Indicates whether the rule should be enabled or disabled. Only enabled rules are used to parse incoming traps and syslogs.</td>
</tr>
<tr>
<td>example syslogMessage</td>
<td>(Optional) Valid for syslogs only. The VCB uses this example syslog message to check the correctness of the regular expression that is automatically created by the VCB based on the user input.</td>
</tr>
<tr>
<td>variable arguments</td>
<td>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 22-27.</td>
</tr>
</tbody>
</table>

**Note**

For the list of global options, see Global Command Options, page 18-21.
Using the CLI to Customize Events

Chapter 20  Managing Additional Event Types Using the VCB

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>No such template name in the templates file.</td>
</tr>
<tr>
<td>211</td>
<td>Event template file not found.</td>
</tr>
<tr>
<td>212</td>
<td>Only one template can be selected from each template category.</td>
</tr>
</tbody>
</table>

**Table 20-15  Error Codes — vcb eventparsingrules modify**

For the list of general VCB error codes, see General Error Codes, page 18-22.

**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.

**Synopsis**

```
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 20-16  Options and Arguments — vcb eventparsingrules delete**

<table>
<thead>
<tr>
<th>Option</th>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>group repository hive</td>
<td>The hive from which to remove the event parsing rule.</td>
</tr>
<tr>
<td></td>
<td>rulename ruleName</td>
<td>The rule to delete.</td>
</tr>
</tbody>
</table>

For the list of global options, see Global Command Options, page 18-21.
Error Codes

Table 20-17  Error Codes—vcb eventparsingrules delete

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>222</td>
<td>Parsing rules repository not found.</td>
</tr>
<tr>
<td>241</td>
<td>No such rule name in the site.xml.</td>
</tr>
</tbody>
</table>

Note For the list of general VCB error codes, see General Error Codes, page 18-22.

vcb eventpattern Commands

Use the following vcb eventpattern commands to create, view, modify, and delete event patterns:

- vcb eventpattern add, page 20-28
- vcb eventpattern view, page 20-29
- vcb eventpattern modify, page 20-31
- vcb eventpattern delete, page 20-32

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.

Synopsis

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 20-18  Options and Arguments—vcb eventpattern add

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>patternid patternId</td>
<td>(Optional) (Recommendation: do not provide.) Unique integer to identify the supported event to VNEs. If not provided, VCB generates this number automatically.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>group parsing rules hive</td>
<td>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.</td>
</tr>
<tr>
<td>repository parsing rules repository hive</td>
<td>The trap or syslog repository where the actual parsing rules are defined.</td>
</tr>
<tr>
<td></td>
<td>Note: Enter the same hive that was specified when creating parsing rules registrations using the vcb eventparsingrules add command.</td>
</tr>
<tr>
<td>rulename rulename</td>
<td>String that is used as a key name for event rule definition.</td>
</tr>
<tr>
<td></td>
<td>Note: Enter exactly the same string as the one that was specified when creating parsing rules registrations using the vcb eventparsingrules add command.</td>
</tr>
</tbody>
</table>

Table 20-19  Error Codes—vcb event pattern add

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>221</td>
<td>Parsing rules hive not found.</td>
</tr>
<tr>
<td>222</td>
<td>Parsing rules repository not found.</td>
</tr>
</tbody>
</table>

Note: For the list of general VCB error codes, see General Error Codes, page 18-22.

Error Codes

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.
Using the CLI to Customize Events

Chapter 20 Managing Additional Event Types Using the VCB

Synopsis

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 20-20 Options and Arguments—vcb eventpattern view

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>group parsingrules hive</td>
<td>The parsing-rules filename used by the NE type or scheme.</td>
</tr>
<tr>
<td>rulename rule name</td>
<td>Unique string that represents the event parsing rules.</td>
</tr>
<tr>
<td>Tip</td>
<td>Enter all as the rule name to list all event parsing rules defined in the repository.</td>
</tr>
</tbody>
</table>
Using the CLI to Customize Events

vcb eventpattern modify

Use the `vcb eventpattern modify` command to modify the pointer to the parsing rules.

**Synopsis**

```plaintext
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**

```plaintext
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.

**Options**

<table>
<thead>
<tr>
<th>Option Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>patternid <code>patternId</code></td>
<td>Unique integer to identify the supported event to a VNE.</td>
</tr>
<tr>
<td>group <code>parsing rules hive</code></td>
<td>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.</td>
</tr>
<tr>
<td>repository <code>parsing rules repository hive</code></td>
<td>(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 <code>vcb eventparsingrules add</code> command.</td>
</tr>
<tr>
<td>rulename <code>ruleName</code></td>
<td>String that is used as a key name for event rule definition. <strong>Note</strong> Enter exactly the same string as the one that was specified when creating parsing rules registrations using the <code>vcb eventparsingrules add</code> command.</td>
</tr>
</tbody>
</table>

**Error Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>221</td>
<td>Parsing rules hive not found.</td>
</tr>
<tr>
<td>222</td>
<td>Parsing rules repository not found.</td>
</tr>
<tr>
<td>271</td>
<td>Pattern with ID not found</td>
</tr>
</tbody>
</table>

**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.
Synopsis

```
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>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>patternid pattern ID</td>
<td>Unique integer to identify the supported event to a VNE.</td>
</tr>
<tr>
<td>group parsing rules hive</td>
<td>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.</td>
</tr>
</tbody>
</table>

Error Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>221</td>
<td>Parsing rules hive not found.</td>
</tr>
<tr>
<td>271</td>
<td>Pattern with ID not found</td>
</tr>
</tbody>
</table>

**Note**

For the list of global options, see **Global Command Options, page 18-21**.

**Note**

For the list of general VCB error codes, see **General Error Codes, page 18-22**.

vcb eventarg Commands

vcb eventarg view

Use the `vcb eventarg view` command to display event parsing rule arguments and descriptions.
Chapter 20  Managing Additional Event Types Using the VCB

Synopsis

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 (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 18-21.

Errors that occur in the server are not as interactive and 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 Testing and Certifying Event Customizations, page 20-12.

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 and restart the AVM (not just the VNE).

runRegTool.sh -gs localhost set 127.0.0.1 avm<avmid>/services/logger/log4j.category.com.sheer.metrocentral.framework.eventapplication.eventcorrelation.SendAlarmMessageUtil DEBUG
runRegTool.sh -gs localhost set 127.0.0.1 avm<avmid>/services/logger/log4j.category.com.sheer.metrocentral.framework.eventmanager.EventManager DEBUG
runRegTool.sh -gs localhost set 127.0.0.1 avm<avmid>/services/logger/log4j.category.com.sheer.metrocentral.framework.eventapplication.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 rule name (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 20-30). 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.GenericSyslogTypeFilterApplication]

# 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.
Troubleshooting Event Customization (CLI)

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 \texttt{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.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
Step & 9 \tabularnewline
\hline
\end{tabular}
\caption{Troubleshooting Event Customization (CLI)}
\end{table}

\textbf{Step 9} Restart the AVM and repeat the above steps until all errors are resolved and the event is parsed correctly and the Prime Network event is generated as expected.
Deploying and Maintaining Extensions in a Prime Network Production System

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

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. See Viewing VCB Registry Customizations, page 21-1.
2. Export the extensions from the Prime Network gateway in the lab to the VcbImportCommands.sh file, located in NETWORKHOME/Main. See Exporting VCB Registry Customizations, page 21-3.
3. Copy the VcbImportCommands.sh file to the Prime Network gateway in the production environment, optionally, remove any extensions that you do not want to deploy from the script, and run the script. See Importing VCB Registry Customizations, page 21-3.
4. Restart Prime Network.

To roll back all VCB extensions, see Deleting VCB Registry Customizations, page 21-3.

Viewing VCB Registry Customizations

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.

```
ana371@servername [~/Main/registry/ConfigurationFiles]% vcb sitechanges view -user root -password admin

>>>>>>>>>>>>>>>>>>>> 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

<table>
<thead>
<tr>
<th>RuleName</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STACKWISE-MEMBER-STATUS</td>
<td>STACKWISE-MEMBER-STATUS</td>
</tr>
</tbody>
</table>

total rows in report: 1

Hive Name:cisco-syslog-repository

<table>
<thead>
<tr>
<th>RuleName</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FWSM-4-109022</td>
<td>FWSM-4-109022</td>
</tr>
<tr>
<td>FWSM-5-713131</td>
<td>FWSM-5-713131</td>
</tr>
</tbody>
</table>

total rows in report: 2

UVne

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
Exporting VCB Registry Customizations

To export registry changes made by the VCB from the site.xml file, enter the following command:

```
vcb sitechanges export -user username -password password
```

This command exports VCB customizations from site.xml to a VCB command script file, VcbImportCommands.sh, located in `NETWORKHOME/Main`.

Importing VCB Registry Customizations

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>After you export registry changes (see Exporting VCB Registry Customizations, page 21-3), copy the VcbImportCommands.sh file to the <code>NETWORKHOME/Main</code> folder on the Prime Network gateway on which you want to import the customizations.</td>
</tr>
<tr>
<td>2</td>
<td>(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 <code>$VCBPATH</code> as shown in this example:</td>
</tr>
<tr>
<td></td>
<td><code>$VCBPATH eventpattern add -rulename L2-DWDM-3-FATAL_2 -group cisco-syslog-ipcore-parsing-rules -repository cisco-router-iox-syslog-repository -user $USER -password $PASS &gt;&gt; &quot;$VCT_IMPORT_SCRIPT_LOG&quot;</code></td>
</tr>
<tr>
<td>3</td>
<td>Change permissions on the script to ensure that it is executable, by entering a command such as this one:</td>
</tr>
<tr>
<td></td>
<td><code>chmod 755 VcbImportCommands.sh</code></td>
</tr>
<tr>
<td>4</td>
<td>Run the script from the <code>NETWORKHOME/Main</code> folder, by entering this command:</td>
</tr>
<tr>
<td></td>
<td><code>VcbImportCommands.sh -user root -password admin</code></td>
</tr>
</tbody>
</table>

Deleting VCB Registry Customizations

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Create a script file, VcbDeleteCommands.sh, in <code>NETWORKHOME/Main</code> by entering this command:</td>
</tr>
<tr>
<td></td>
<td><code>vcb sitechanges delete -user username -password password</code></td>
</tr>
<tr>
<td>2</td>
<td>(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 <code>$VCBPATH</code> as shown in this example:</td>
</tr>
<tr>
<td></td>
<td><code>$VCBPATH eventpattern delete -group cisco-syslog-ipcore-parsing-rules -patternid 5001 -user $USER -password $PASS &gt;&gt; &quot;$VCT_IMPORT_SCRIPT_LOG&quot;</code></td>
</tr>
</tbody>
</table>
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 root -password admin
```

Related Topic
- vcb sitechanges, page 18-22

Rolling Back VCB Customizations

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.
VCB Template Reference

Use the information in this chapter to determine which template best matches the device, module, or event that you want to manage with Prime Network. Information in this chapter is also useful when you test the customizations that you have made.

Topics include:
- U-VNE Templates, page 22-1
- Module Templates, page 22-10
- Event Templates, page 22-24

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 VNE. The advantage of the U-VNE—created using the VCB and the GenericUVNE template—over the Generic 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:
- Generic VNE, U-VNE, and Developed VNE Comparison, page 18-5
- Advantages and Limitations of U-VNEs, page 18-3
- GenericUVNE Template, page 22-1

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. The generic U-VNE is very similar to the generic 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 Generic U-VNE—Supported Traps, page 22-4. Using the VCB, however, you can configure additional traps and syslog recognition for a generic U-VNE.

The generic U-VNE models NEs using SNMP MIB-II, which is the most generic and widely used management interface. The generic 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 generic U-VNE.

Table 22-1 summarize the features and advantages of the GenericUVNE template.

<table>
<thead>
<tr>
<th>Features</th>
<th>Advantages</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Same physical and logical inventory as a Prime Network generic VNE.</td>
<td>When compared with a generic VNE, a generic U-VNE provides:</td>
<td>Event association to Managed Element only</td>
</tr>
<tr>
<td>• Applicable to Cisco and non-Cisco NEs.</td>
<td>• Simplified trap and syslog recognition (using the VCB).</td>
<td></td>
</tr>
<tr>
<td>• User-defined device type attributes:</td>
<td>• Application of soft property and activation scripts to a specific generic U-VNE (as opposed to a device type).</td>
<td></td>
</tr>
<tr>
<td>– Device category—Determines the icon that is displayed.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>– Element type.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• Event recognition, enabling Prime Network to forward events from unsupported devices to OSS applications.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following terminology is used in the tables that describe the Generic U-VNE:

• Expedite Legend—The Expedites column in the service event (service alarm) 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.

• Information Model Object (IMO)—For more information about IMOs, see the Cisco Prime Network 3.8 Reference Guide.

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:

• Generic U-VNE—Physical Inventory Model, page 22-3
• Generic U-VNE—Logical Inventory Model, page 22-3
• Generic U-VNE—Supported Traps, page 22-4
• Generic U-VNE—Supported Events, page 22-5
• Generic U-VNE—Limitations, page 22-5
• Generic U-VNE—Supported Topologies, page 22-5
• Generic U-VNEs—Supported Technologies, page 22-5
• Generic U-VNE—Supported Service Events, page 22-10

**Generic U-VNE—Physical Inventory Model**

The generic U-VNE uses a static model for the device chassis. The rest of the physical inventory is modeled using the ifTable. Since modules are not modeled, the generic U-VNE creates a single generic module on which all of the physical interfaces reside.

Table 22-2 describes which MIB tables are used to model the physical inventory components that are supported by the generic U-VNE.

<table>
<thead>
<tr>
<th>Logical Component</th>
<th>MIB Table</th>
<th>Columns/Tables Used For Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interfaces</td>
<td>ifTable</td>
<td>• ifDescr</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ifType</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ifOperStatus</td>
</tr>
<tr>
<td>Ports</td>
<td></td>
<td>• ifOperStatus and ifAdminStatus</td>
</tr>
<tr>
<td>Port status</td>
<td>ifTable</td>
<td>• ifSpeed</td>
</tr>
<tr>
<td>Port speed</td>
<td>ifTable</td>
<td>• ifPhysAddress (Ethernet ports)</td>
</tr>
<tr>
<td>MAC address</td>
<td>ifTable</td>
<td></td>
</tr>
</tbody>
</table>

---

**Note**

Certain general properties on the managed element, such as system description, are modeled using the RFC1213-MIB.

**Generic U-VNE—Logical Inventory Model**

Table 22-3 describes which MIB tables are used to model the logical inventory components that are supported by the generic U-VNE. Attributes in Table 22-3 are taken from MIB-II.

<table>
<thead>
<tr>
<th>Logical Component</th>
<th>MIB Table</th>
<th>Columns/Tables Used For Modeling</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP Interfaces</td>
<td>ipAddrTable</td>
<td>• ipAdEntIfIndex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ipAdEntNetMask</td>
</tr>
<tr>
<td>ARP table</td>
<td>ipNetToMediaTable</td>
<td>• ipNetToMediaPhysAddress</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ipNetToMediaType</td>
</tr>
</tbody>
</table>
Chapter 22      VCB Template Reference

Table 22-3  MIBs Used for Logical Inventory Model of Generic U-VNE (continued)

<table>
<thead>
<tr>
<th>Logical Component</th>
<th>MIB Table</th>
<th>Columns/Tables Used For Modeling</th>
</tr>
</thead>
</table>
| Routing table     | ipRouteTable       | • ipRouteDest
|                   |                    | • ipRouteIfIndex
|                   |                    | • ipRouteIfIndexHop
|                   |                    | • ipRouteNextHop
|                   |                    | • ipRouteType
|                   |                    | • ipRouteMask
| Bridging table    | dot1dTpFdbTable    | —                                                         |
| Default bridge    | dot1dBridge        | • dot1dBaseBridgeAddress                                 |
|                   |                    | • dot1dBaseType                                          |

Generic U-VNE—Supported Traps

The generic U-VNE can parse the standard MIB-II and Bridge-MIB traps listed in Table 22-4.

Table 22-4  Supported Traps for Generic U-VNE

<table>
<thead>
<tr>
<th>Standard MIB-II Traps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>authenticationFailure</td>
<td>mplsTunnelReoptimized</td>
</tr>
<tr>
<td>bgpBackwardTransition</td>
<td>mplsTunnelRerouted</td>
</tr>
<tr>
<td>bgpEstablished</td>
<td>mplsTunnelUp</td>
</tr>
<tr>
<td>coldStart</td>
<td>ospfIfAuthFailure</td>
</tr>
<tr>
<td>entConfigChange</td>
<td>ospfIfConfigError</td>
</tr>
<tr>
<td>linkDown</td>
<td>ospfIfRxBadPacket</td>
</tr>
<tr>
<td>linkUp</td>
<td>ospfIfStateChange (down)</td>
</tr>
<tr>
<td>mplsL3VpnVrfDown</td>
<td>ospfIfStateChange (up)</td>
</tr>
<tr>
<td>mplsL3VpnVrfRouteMaxThreshExceeded</td>
<td>ospfMaxAgeLsa</td>
</tr>
<tr>
<td>mplsL3VpnVrfRouteMidThreshExceeded</td>
<td>ospfNbrStateChange (down)</td>
</tr>
<tr>
<td>mplsL3VpnVrfUp</td>
<td>ospfNbrStateChange (up)</td>
</tr>
<tr>
<td>mplsLdpInitSessionThresholdExceeded</td>
<td>ospf-if-packet-retransmit</td>
</tr>
<tr>
<td>mplsLdpSessionDown</td>
<td>ospfOriginateLsa</td>
</tr>
<tr>
<td>mplsLdpSessionUp</td>
<td>ospfTxRetransmit</td>
</tr>
<tr>
<td>mplsTunnelDown</td>
<td>warmStart</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bridge-MIB Traps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>dot1dBaseBridgeAddress</td>
<td>dot1dBaseType</td>
</tr>
</tbody>
</table>

The generic U-VNE can identify traps, but it cannot correlate them. This is because the generic 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
generic U-VNE, Prime Network can identify the Tunnel Down trap, but it cannot perform correlation on
the trap. The reason is that the generic 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 the generic U-VNE, event association is always to the Managed Element.

**Generic U-VNE—Supported Events**

The generic U-VNE supports the service events listed in Table 22-5.

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Supported</th>
<th>Expedited</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device Unreachable</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Discard Packets</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Dropped Packets</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Port Flapping</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Port Down</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

**Generic U-VNE—Limitations**

The generic U-VNE 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 the generic
U-VNE is dependent on the vendor’s adherence to general network management standards.

**Generic U-VNE—Supported Topologies**

The generic U-VNE supports the topologies listed in Table 22-6.

<table>
<thead>
<tr>
<th>Topology Type</th>
<th>Link Type</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>Ethernet</td>
<td>Y</td>
</tr>
<tr>
<td>Physical Layer</td>
<td>Ethernet</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Generic U-VNEs—Supported Technologies**

The following sections list the objects and attributes that are recognized on generic U-VNEs per
technology:

- IP, page 22-6
- Ethernet (IEEE 802.3), page 22-7
- Base Logical Components, page 22-7
- Common, page 22-8
Table 22-7 lists the IP attribute support on generic U-VNEs.

**Note** Table 22-7 includes the supported technologies only.

### Table 22-7  IP Attribute Support on Generic U-VNEs

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMO Name—IIIInterface</td>
<td></td>
</tr>
<tr>
<td>IP Address</td>
<td>Y</td>
</tr>
<tr>
<td>Subnetwork Mask</td>
<td>Y</td>
</tr>
<tr>
<td>IP Interface Addresses Array</td>
<td></td>
</tr>
<tr>
<td>Interface Name</td>
<td></td>
</tr>
<tr>
<td>Interface Description</td>
<td>Y</td>
</tr>
<tr>
<td>IP Interface State</td>
<td>Y</td>
</tr>
<tr>
<td>OSPF Interface Cost</td>
<td></td>
</tr>
<tr>
<td>Broadcast Address</td>
<td></td>
</tr>
<tr>
<td>MTU</td>
<td></td>
</tr>
<tr>
<td>Lookup Method</td>
<td></td>
</tr>
<tr>
<td>Address Resolution Type</td>
<td></td>
</tr>
<tr>
<td>ARP Timeout</td>
<td></td>
</tr>
<tr>
<td>Secured ARP</td>
<td></td>
</tr>
<tr>
<td>ICMP Mask Reply</td>
<td></td>
</tr>
<tr>
<td>IGMP Proxy</td>
<td></td>
</tr>
<tr>
<td>HSRP Groups</td>
<td></td>
</tr>
<tr>
<td>IP Multiplexing Table</td>
<td></td>
</tr>
<tr>
<td>IANA Type</td>
<td></td>
</tr>
<tr>
<td>Containing CTPs</td>
<td></td>
</tr>
<tr>
<td>Contained CTPs</td>
<td></td>
</tr>
<tr>
<td>IMO Name—IIRoutingEntity</td>
<td></td>
</tr>
<tr>
<td>Routing Table</td>
<td>Y</td>
</tr>
<tr>
<td>ARP Entity</td>
<td>Y</td>
</tr>
<tr>
<td>Routing Table Changes</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Y</td>
</tr>
<tr>
<td>Logical Sons</td>
<td>Y</td>
</tr>
<tr>
<td>IMO Name—IIRoutingEntry</td>
<td></td>
</tr>
<tr>
<td>Destination IP Subnet</td>
<td>Y</td>
</tr>
</tbody>
</table>
Table 22-7  IP Attribute Support on Generic U-VNEs (continued)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Hop IP Address</td>
<td>Y</td>
</tr>
<tr>
<td>Type</td>
<td>Y</td>
</tr>
<tr>
<td>Routing Protocol Type</td>
<td>Y</td>
</tr>
<tr>
<td>Outgoing Interface Name</td>
<td>Y</td>
</tr>
<tr>
<td>IMO Name—IARPEntity</td>
<td></td>
</tr>
<tr>
<td>ARP Table</td>
<td>Y</td>
</tr>
<tr>
<td>IMO Name—IARPEntry</td>
<td></td>
</tr>
<tr>
<td>IP Address</td>
<td>Y</td>
</tr>
<tr>
<td>MAC Address</td>
<td>Y</td>
</tr>
<tr>
<td>Port</td>
<td>Y</td>
</tr>
<tr>
<td>Entry Type</td>
<td>Y</td>
</tr>
</tbody>
</table>

**Ethernet (IEEE 802.3)**

Table 22-8 lists the Ethernet (IEEE 802.3) attribute support on generic U-VNEs.

Table 22-8  Ethernet (IEEE 802.3) Attribute Support on Generic U-VNEs

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMO Name—IEthernet</td>
<td></td>
</tr>
<tr>
<td>MAC Address</td>
<td>Y</td>
</tr>
<tr>
<td>Duplex Mode</td>
<td></td>
</tr>
<tr>
<td>Output Flow Control</td>
<td></td>
</tr>
<tr>
<td>Input Flow Control</td>
<td></td>
</tr>
<tr>
<td>IANA Type</td>
<td></td>
</tr>
<tr>
<td>Containing CTPs</td>
<td></td>
</tr>
<tr>
<td>Contained CTPs</td>
<td></td>
</tr>
<tr>
<td>Port Type</td>
<td></td>
</tr>
</tbody>
</table>

**Base Logical Components**

Table 22-9 lists the base logical attribute support on generic U-VNEs.

Note  Table 22-9 includes the supported technologies only.
### Table 22-9  Base Logical Components Attribute Support on Generic U-VNEs

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>** IMO Name—IMangedElement**</td>
<td></td>
</tr>
<tr>
<td>IP Address</td>
<td>Y</td>
</tr>
<tr>
<td>Communication State</td>
<td>Y</td>
</tr>
<tr>
<td>Investigation State</td>
<td>Y</td>
</tr>
<tr>
<td>Element Category</td>
<td>Y</td>
</tr>
<tr>
<td>Element Type and Key</td>
<td>Y</td>
</tr>
<tr>
<td>Device Name</td>
<td>Y</td>
</tr>
<tr>
<td>System Name</td>
<td>Y</td>
</tr>
<tr>
<td>System Description</td>
<td>Y</td>
</tr>
<tr>
<td>Up Time</td>
<td>Y</td>
</tr>
<tr>
<td>Software Version</td>
<td>Y</td>
</tr>
<tr>
<td>Vendor Identity</td>
<td></td>
</tr>
<tr>
<td>Memory and CPU Usage</td>
<td></td>
</tr>
<tr>
<td>DRAM Free</td>
<td></td>
</tr>
<tr>
<td>DRAM Used</td>
<td></td>
</tr>
<tr>
<td>Flash Device Size</td>
<td></td>
</tr>
<tr>
<td>NVRAM Size</td>
<td></td>
</tr>
<tr>
<td>Processor DRAM</td>
<td></td>
</tr>
<tr>
<td>Sys Contact</td>
<td></td>
</tr>
<tr>
<td>Sys Location</td>
<td></td>
</tr>
<tr>
<td>Serial Number</td>
<td></td>
</tr>
<tr>
<td>File Systems</td>
<td></td>
</tr>
</tbody>
</table>

**IMO Name—ISystemService**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Up Time</td>
<td></td>
</tr>
</tbody>
</table>

**Common**

Table 22-10 lists the common attribute support on generic U-VNEs.

**Note**  Table 22-10 includes the supported technologies only.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Supported</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>IMO Name—IPhysicalLayer</strong></td>
<td></td>
</tr>
<tr>
<td>Media Type</td>
<td></td>
</tr>
<tr>
<td>Clocking Source</td>
<td></td>
</tr>
<tr>
<td>Maximum Speed</td>
<td>Y</td>
</tr>
<tr>
<td>Is Internal Port</td>
<td></td>
</tr>
<tr>
<td>Discarded Bandwidth</td>
<td></td>
</tr>
<tr>
<td>Dropped Bandwidth</td>
<td></td>
</tr>
<tr>
<td>Input Bandwidth</td>
<td></td>
</tr>
<tr>
<td>Output Bandwidth</td>
<td></td>
</tr>
<tr>
<td>Discarded and Received Input Data Counters</td>
<td>Y</td>
</tr>
<tr>
<td>Dropped and Forward Output Data Counters</td>
<td>Y</td>
</tr>
<tr>
<td>Administrative Status</td>
<td>Y</td>
</tr>
<tr>
<td>Operational Status</td>
<td>Y</td>
</tr>
<tr>
<td>Last Changed</td>
<td>Y</td>
</tr>
<tr>
<td>IANA Type</td>
<td></td>
</tr>
<tr>
<td>Containing CTPs</td>
<td></td>
</tr>
<tr>
<td>Contained CTPs</td>
<td></td>
</tr>
<tr>
<td>Port Alias</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Sending Alarms</td>
<td></td>
</tr>
<tr>
<td>Connector Description</td>
<td></td>
</tr>
<tr>
<td>Part ID</td>
<td></td>
</tr>
<tr>
<td>Connector Serial Num</td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td></td>
</tr>
<tr>
<td>Managed</td>
<td></td>
</tr>
<tr>
<td><strong>IMO Name—IBridgeEntry</strong></td>
<td></td>
</tr>
<tr>
<td>Destination MAC</td>
<td>Y</td>
</tr>
<tr>
<td>Outgoing Interface</td>
<td>Y</td>
</tr>
</tbody>
</table>
Module Templates

Note: 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.

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. For example, typical port layers for an OC3 ATM card are:

- Layer 0—Fiber Optic
- Layer 1—OC-3
- Layer 2—ATM

Use module templates to enable developed VNEs to recognize new modules.

Note: You cannot add modules to generic U-VNEs.

Module templates ensure that each port is modeled with the correct port layer information based on the ifType obtained from the SNMP MIB output.

This section contains the following topics:

- Module Groups and Module Specification Files, page 22-10
- Module Templates by Technology, page 22-14

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 Commands for Managing Modules, page 19-10.)
- 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 22-12 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 22-12  Module Group Summary for Standard Modules

<table>
<thead>
<tr>
<th>Module Group</th>
<th>Technologies</th>
<th>Template Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>ciscophysicals</td>
<td>Ethernet (Fixed), page 22-14</td>
<td>• A10GigaEthernet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ethernet-default-over-optic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ethernetDefault</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• gigaEthernet</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• EthernetChannelSwitchDefault</td>
</tr>
<tr>
<td></td>
<td>Ethernet (Multiloader), page 22-15</td>
<td>• 10Gigaethernet-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Gigaethernet-Fiber</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• GE-fiberoptic-ethernet-default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ethernetDefault-RJ45-or-Fiber2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• E1orGigabitTechnology2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• GE-over-OC12-pos-default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• GE-over-OC3-pos-default3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ethernet-or-oc-pos-default3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ethernet-or-OC12-pos-default3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• pos-default3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ethernet-or-oc48-pos-default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• pos-default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DWDMA10GigaEthernet</td>
</tr>
<tr>
<td>POS (Fixed), page 22-16</td>
<td></td>
<td>• PPPdefaultOC48</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PPPdefault</td>
</tr>
<tr>
<td>POS (Multiloader), page 22-17</td>
<td></td>
<td>• POS-OC3-default</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PPPdefaultOC12</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PPPdefaultOC192</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PPPdefaultOC3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• PPPdefaultOC768</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• DWDMOC768</td>
</tr>
</tbody>
</table>
### Table 22-12  Module Group Summary for Standard Modules (continued)

<table>
<thead>
<tr>
<th>Module Group</th>
<th>Technologies</th>
<th>Template Names</th>
</tr>
</thead>
</table>
| ciscophysicalspec2 (continued) | Channelized T1/E1 (Fixed), page 22-18 | • RJ45-T1E1Channelized  
• T1E1Channelized  
• T1E1Channelized-ATMorCEM  
• E3Default  
• E3Loader  
• E1Channelized  
• E1Default |
| Channelized OCXX (Fixed), page 22-18 | | • ChannelizedOC3  
• ChannelizedOC12  
• ChannelizedOC12xx  
• ChannelizedOCxx |
| ATM (Fixed), page 22-19 | | • atmDefault  
• atmOverOC12  
• atm-over-e3ds3  
• T1E1_ATM-IMA  
• ds1Default  
• ds3Default  
• adslDefault |
| ATM (Multiloader), page 22-19 | | • T3Channelized  
• T3Loader  
• layer2-over-ds1  
• layer2-over-ds3  
• layer2-over-ds3-over-bnc  
• pppOverDS3Default  
• layer2-over-e1  
• HSSIDefault |
| Multitechnology, page 22-20 | | • 36xxMultiTechnologiesModuleDefault  
• 8xxMultiTechnologiesModuleDefault  
• MultiTechnologiesModuleLayers  
• MultiTechnologiesModuleDefault2 |
| Serial, page 22-21 | | • PPPwithRJ11  
• multichannelDefault  
• serialPPPDefault |
| ISDN, page 22-22 | | • ds1T1Default  
• BRIDefault |
Module Templates by Technology

This section presents module templates organized by technology:

- Ethernet (Fixed), page 22-14
- Ethernet (Multiloader), page 22-15
- POS (Fixed), page 22-16
- POS (Multiloader), page 22-17
- Channelized T1/E1 (Fixed), page 22-18
- Channelized OCXX (Fixed), page 22-18
- ATM (Fixed), page 22-19
- ATM (Multiloader), page 22-19
- Multitechnology, page 22-20
- Serial, page 22-21
- ISDN, page 22-22
- Generic, page 22-22
- Ethernet (Cisco Catalyst 3400), page 22-23
- Ethernet (Cisco Catalyst), page 22-23

**Ethernet (Fixed)**

Table 22-13 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 22-13  Module Templates—Ethernet (Fixed)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>A10GigaEthernet</td>
<td>Fiber optic</td>
<td>EthernetCSMA/CD</td>
<td>10 Gigabit Ethernet</td>
<td>• WS-SUP32-10GE-3B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 7600-ES+2TG</td>
</tr>
<tr>
<td>ethernet-default-over-optic</td>
<td>Fiber optic</td>
<td>EthernetCSMA/CD</td>
<td></td>
<td>• WS-6700-DFC3B</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Ethernet</td>
<td>• WS-6700-DFC3BXL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fast Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>ethernetDefault</td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td></td>
<td>• 8FE-TX-RJ45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Ethernet</td>
<td>• SPA-8X1FE-TX-V2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fast Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>gigaEthernet</td>
<td>Fiber optic</td>
<td>EthernetCSMA/CD</td>
<td>Gigabit Ethernet</td>
<td>• WS-X4624-SFP-E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• 7600-ES+3C</td>
</tr>
<tr>
<td>EthernetChannelSwitchDefault</td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td>EtherChannel</td>
<td>NM-16ESW</td>
</tr>
</tbody>
</table>

Table 22-13 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 22-14 also support POS or DWDM ports. (See the footnotes for Table 22-14.)

Table 22-14  Module Templates—Ethernet (Multiloader)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>10Gigaethernet-Gigaethernet-Fiber1</td>
<td>• Fiber optic</td>
<td>EthernetCSMA/CD</td>
<td>10 Gigabit Ethernet</td>
<td>• 76-ES+XC-20G3C</td>
</tr>
<tr>
<td></td>
<td>• RJ45</td>
<td></td>
<td></td>
<td>• WS-X45-SUP6-E</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td>• WS-X4606-X2-E</td>
</tr>
<tr>
<td>GE-fiberoptic-ethernet-default2</td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td></td>
<td>catalyst375024ME</td>
</tr>
<tr>
<td></td>
<td>Fiber optic</td>
<td></td>
<td>• Ethernet</td>
<td>(cevModuleCat375024M)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fast Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>ethernetDefault-RJ45-or-Fiber2</td>
<td>• Fiber optic</td>
<td>EthernetCSMA/CD</td>
<td></td>
<td>• WS-X4232-RJ-XX</td>
</tr>
<tr>
<td></td>
<td>• RJ45</td>
<td></td>
<td>• Ethernet</td>
<td>• WS-X4524-GB-RJ45V</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fast Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>E1orGigabitTechnology2</td>
<td>• RJ45</td>
<td>EthernetCSMA/CD</td>
<td></td>
<td>Motherboard for 2941</td>
</tr>
<tr>
<td></td>
<td>Fiber optic</td>
<td></td>
<td>• Ethernet</td>
<td>(cevCpu2941)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fast Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
</tbody>
</table>
Table 22-14  
**Module Templates—Ethernet (Multiloader) (continued)**

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>GE-over-OC12-pos-default³</td>
<td>Fiber optic</td>
<td>OC12</td>
<td>• PPP</td>
<td>OSM-4OC12-POS-SI+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EthernetCSMA/CD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>GE-over-OC3-pos-default³</td>
<td>Fiber optic</td>
<td>OC3</td>
<td>• PPP</td>
<td>OSM-4OC3-POS-SI+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td>Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>ethernet-or-oc-pos-default³</td>
<td>Fiber optic</td>
<td>OC3</td>
<td>• PPP</td>
<td>OSM-2+4GE-WAN+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EthernetCSMA/CD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fast Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>ethernet-or-OC12-pos-default³</td>
<td>Fiber optic</td>
<td>OC12</td>
<td>• PPP</td>
<td>• OSM-2OC12-POS-SI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td>• OSM-2OC12-POS-SI+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EthernetCSMA/CD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fast Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>ethernet-or-oc48-pos-default</td>
<td>Fiber optic</td>
<td>OC48</td>
<td>• PPP</td>
<td>OSM-1OC48-POS-SI+</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>EthernetCSMA/CD</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Fast Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• 10 Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td>DWDMA10GigaEthernet⁴</td>
<td>Fiber optic</td>
<td>• DWDM</td>
<td>10 Gigabit Ethernet</td>
<td>• 76-ES+XT-2TG3CXL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• EthernetCSMA/CD</td>
<td></td>
<td>• 76-ES+XT-4TG3C</td>
</tr>
</tbody>
</table>

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 22-15 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 22-15  Module Templates—POS (Fixed)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Maximum Transmission Rate (Mbps) Supported</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPPdefaultOC48</td>
<td>2488.32</td>
<td>Fiber optic</td>
<td>OC48</td>
<td>• PPP</td>
<td>• SFP-OC48-IR1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td>• gsr-e-qoc48-sm-lr-sc</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td>• 16OC48-POS/DPT</td>
</tr>
<tr>
<td>PPPdefault</td>
<td>155.52</td>
<td>Fiber optic</td>
<td>OC3</td>
<td>• PPP</td>
<td>• GSR-SFC6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td>• GSR-CSC</td>
</tr>
</tbody>
</table>

POS (Multiloader)

Table 22-16 lists templates for POS modules with multiple Layer 2 options.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table 22-16  Module Templates—POS (Multiloader)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Maximum Transmission Rate (Mbps) Supported</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS-OC3-default</td>
<td>155.52</td>
<td>Fiber optic</td>
<td>OC3</td>
<td>• PPP</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame Relay</td>
<td></td>
</tr>
<tr>
<td>PPPdefaultOC12</td>
<td>622.08</td>
<td>Fiber optic</td>
<td>OC12</td>
<td>• PPP</td>
<td>SPA-8XOC12-POS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame Relay</td>
<td></td>
</tr>
<tr>
<td>PPPdefaultOC192</td>
<td>9,953.28</td>
<td>Fiber optic</td>
<td>OC192</td>
<td>• PPP</td>
<td>SPA-OC192POS-LR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td>PPPdefaultOC3</td>
<td>155.52</td>
<td>Fiber optic</td>
<td>OC3</td>
<td>• PPP</td>
<td>• SFP-OC3-SR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td>• SFP-OC3-IR1</td>
</tr>
<tr>
<td>PPPdefaultOC768</td>
<td>39,813.12</td>
<td>Fiber optic</td>
<td>OC768</td>
<td>• PPP</td>
<td>1OC768-ITU/C</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td>DWDMOC768</td>
<td>39,813.12</td>
<td>Fiber optic</td>
<td>OC768</td>
<td>• PPP</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
</tbody>
</table>
Module Templates

Chapter 22  VCB Template Reference

Channelized T1/E1 (Fixed)

Table 22-17 lists templates for channelized T1/E1 modules where port layers are fixed.

Module Group
These templates are defined in the ciscophysicalspec2 module group.

Table 22-17  Module Templates—Channelized T1/E1 (Fixed)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Maximum Transmission Rate (Mbps) Supported</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>RJ45-T1E1Channelized</td>
<td>1.544</td>
<td>RJ45</td>
<td>T1E1</td>
<td>—</td>
<td>SPA-8XCHT1/E1</td>
</tr>
<tr>
<td>T1E1Channelized</td>
<td>1.544</td>
<td>RJ48</td>
<td>T1E1</td>
<td>—</td>
<td>• NM-2CE1T1-PRI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• PA-MC-4T1</td>
</tr>
<tr>
<td>T1E1Channelized-ATM or CEM1</td>
<td>1.544</td>
<td>RJ48</td>
<td>T1E1</td>
<td>—</td>
<td>HWIC-4T1/E1</td>
</tr>
<tr>
<td>E3Default</td>
<td>44.736</td>
<td>BNC</td>
<td>DS3</td>
<td>—</td>
<td>ESR-8E3/DS3</td>
</tr>
<tr>
<td>E3Loader</td>
<td>44.736</td>
<td>BNC</td>
<td>DS3</td>
<td>—</td>
<td></td>
</tr>
<tr>
<td>E1Channelized</td>
<td>1.544</td>
<td>RJ48</td>
<td>E1</td>
<td>—</td>
<td>PA-8CE1</td>
</tr>
<tr>
<td>E1Default</td>
<td>1.544</td>
<td>RJ45</td>
<td>E1T1</td>
<td>—</td>
<td>VWIC2-1MFT-T1E1</td>
</tr>
</tbody>
</table>

1. Layer 2—ATM or CEM—is built at runtime.

Channelized OCXX (Fixed)

Table 22-18 lists templates for Optical Carrier modules where port layers are fixed.

Module Group
These templates are defined in the ciscophysicalspec2 module group.

Table 22-18  Module Templates—Channelized OCXX (Fixed)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Maximum Transmission Rate (Mbps) Supported</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChannelizedOC3</td>
<td>155.52</td>
<td>Fiber optic</td>
<td>OC3</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ChannelizedOC12</td>
<td>622.08</td>
<td>Fiber optic</td>
<td>OC12</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>ChannelizedOC12xx</td>
<td>622.08</td>
<td>Fiber optic</td>
<td>OC12</td>
<td>—</td>
<td>SPA-1XCHOC12/DS0</td>
</tr>
<tr>
<td>Note</td>
<td>See CSCtd93353 in CDETs.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ChannelizedOCxx</td>
<td>155.52</td>
<td>Fiber optic</td>
<td>OC3</td>
<td>—</td>
<td>• SPA-1XCHSTM1/OC3</td>
</tr>
<tr>
<td>Note</td>
<td>See CSCtd93353 in CDETs.</td>
<td></td>
<td></td>
<td></td>
<td>• SPA-CHOC3-CE-ATM</td>
</tr>
</tbody>
</table>
ATM (Fixed)

Table 22-19 lists templates for ATM modules where port layers are fixed.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table 22-19  Module Templates—ATM (Fixed)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Maximum Transmission Rate (Mbps) Supported</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>atmDefault</td>
<td>155.52</td>
<td>Fiber optic</td>
<td>OC3</td>
<td>ATM</td>
<td>GSR-SFC12410</td>
</tr>
<tr>
<td>atmOverOC12</td>
<td>—</td>
<td>Fiber optic</td>
<td>OC12</td>
<td>ATM</td>
<td>• SPA-1XOC12-ATM-V2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• SPA-1XOC12-ATM</td>
</tr>
<tr>
<td>atm-over-e3ds3</td>
<td>—</td>
<td>RJ48</td>
<td>DS3</td>
<td>ATM</td>
<td>—</td>
</tr>
<tr>
<td>T1E1_ATM-IMA</td>
<td>1.544</td>
<td>RJ48</td>
<td>T1E1</td>
<td>ATM</td>
<td>—</td>
</tr>
<tr>
<td>ds1Default</td>
<td>—</td>
<td>BNC</td>
<td>DS1</td>
<td>ATM</td>
<td>—</td>
</tr>
<tr>
<td>ds3Default</td>
<td>—</td>
<td>BNC</td>
<td>DS3</td>
<td>ATM</td>
<td>PA-A3-T3</td>
</tr>
<tr>
<td>adslDefault</td>
<td>—</td>
<td>RJ11</td>
<td>ADSL</td>
<td>ATM</td>
<td>• WIC-1SHDSL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• WIC-1ADSL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• WIC-1ADSL-DG</td>
</tr>
</tbody>
</table>

ATM (Multiloader)

Table 22-20 lists templates for ATM modules with multiple Layer 2 options.

Module Group

These templates are defined in the ciscophysicalspec2 module group.

Table 22-20  Module Templates—ATM (Multiloader)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Maximum Transmission Rate (Mbps) Supported</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>T3Channelized¹</td>
<td>—</td>
<td>BNC</td>
<td>DS3</td>
<td>• PPP</td>
<td>• SPA-2XCT3/DS0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td>• SPA-2T3/E3-EC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• ATM</td>
<td></td>
</tr>
<tr>
<td>T3Loader</td>
<td>—</td>
<td>BNC</td>
<td>DS3</td>
<td>• PPP</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
</tbody>
</table>
Table 22-20  Module Templates—ATM (Multiloader) (continued)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Maximum Transmission Rate (Mbps) Supported</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>layer2-over-ds1</td>
<td>1.544</td>
<td>RJ48</td>
<td>DS1</td>
<td>• PPP</td>
<td>VWIC-2MFT-T1-DIR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• ATM</td>
<td></td>
</tr>
<tr>
<td>layer2-over-ds3</td>
<td>44.736</td>
<td>RJ48</td>
<td>DS3</td>
<td>• PPP</td>
<td>• NM-1A-E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td>• NM-1T3/E3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• ATM</td>
<td></td>
</tr>
<tr>
<td>layer2-over-ds3-over-bnc</td>
<td>44.736</td>
<td>BNC</td>
<td>DS3</td>
<td>• PPP</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• ATM</td>
<td></td>
</tr>
<tr>
<td>pppOverDS3Default</td>
<td>44.736</td>
<td>BNC</td>
<td>DS3</td>
<td>• PPP</td>
<td>• copper-6ds3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td>• copper-12ds3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td>• 2DS3-SMB</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• ATM</td>
<td>• NM-4T</td>
</tr>
<tr>
<td>layer2-over-e1</td>
<td>1.544</td>
<td>RJ48</td>
<td>E1</td>
<td>• PPP</td>
<td>NM-1CE1T1-PRI</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• ATM</td>
<td></td>
</tr>
<tr>
<td>HSSIDefault</td>
<td>—</td>
<td>DB50</td>
<td>HSSI</td>
<td>• PPP</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
</tbody>
</table>

1. This module template supports both full and channelized T3 and ATM over T3.

**Multitechnology**

Table 22-21 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 22-21  Module Templates—Multitechnology

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>36xxMultiTechnologiesModule Default¹</td>
<td>RJ45</td>
<td>• EthernetCSMA/CD</td>
<td>• Fast Ethernet</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Serial</td>
<td>• PPP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td>8xxMultiTechnologiesModuleDefault</td>
<td>RJ45</td>
<td>• EthernetCSMA/CD</td>
<td>• Fast Ethernet</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Serial</td>
<td>• PPP</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td>MultiTechnologiesModule Layers²</td>
<td>RJ11</td>
<td>• EthernetCSMA/CD</td>
<td>• Fast Ethernet</td>
<td>OSM-1CHOC12/T3-SI</td>
</tr>
<tr>
<td></td>
<td>RJ45</td>
<td>• DS1</td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RJ48</td>
<td>• E1</td>
<td>• PPP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fiber optic</td>
<td>• OC3</td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB60</td>
<td>• ADSL</td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Serial</td>
<td>• ATM</td>
<td></td>
</tr>
<tr>
<td>MultiTechnologiesModule Default²</td>
<td>RJ11</td>
<td>• EthernetCSMA/CD</td>
<td>• Fast Ethernet</td>
<td>gsr-sfc16-oc192</td>
</tr>
<tr>
<td></td>
<td>RJ45</td>
<td>• DS1</td>
<td>• Gigabit Ethernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RJ48</td>
<td>• E1</td>
<td>• PPP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fiber optic</td>
<td>• OC3</td>
<td>• HDLC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DB60</td>
<td>• ADSL</td>
<td>• Frame relay</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Serial</td>
<td>• ATM</td>
<td></td>
</tr>
</tbody>
</table>

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 22-22 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 22-22  Module Templates—Serial

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPPwithRJ11</td>
<td>RJ11</td>
<td>Serial</td>
<td>PPP</td>
<td>WIC-1AM-V2</td>
</tr>
<tr>
<td>multichannelDefault</td>
<td>RJ48</td>
<td>Serial</td>
<td>PPP</td>
<td>—</td>
</tr>
<tr>
<td>serialPPPDefault</td>
<td>RJ45</td>
<td>RJ48</td>
<td>Fiber optic</td>
<td>DB60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Generic connector</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Serial</td>
<td>PPP</td>
<td>HDLC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Frame relay</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>HWIC-4T</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>NM-2W</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WIC-SERIAL-1T</td>
</tr>
</tbody>
</table>

Table 22-23  Module Templates—ISDN

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Maximum Transmission Rate (kbps) Supported</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>ds1T1Default</td>
<td>—</td>
<td>RJ45</td>
<td>DS1</td>
<td>ISDN</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RJ48</td>
<td>E1</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DB60</td>
<td>EthernetCSMA/CD</td>
<td>ISDN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fiber optic</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BRIDefault</td>
<td>64</td>
<td>RJ45</td>
<td>ISDN layer 1</td>
<td>ISDN layer 2</td>
<td>WIC-1B-U-V2</td>
</tr>
</tbody>
</table>

1. The connector type is modeled based on the type of port.

Generic

Table 22-24 lists generic templates. Use them to configure modules for technologies that Prime Network does not support.

Module Group

These templates are defined in the ciscophysicalspe2 module group.
### Table 22-24  Module Templates—Generic

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSLineDefault¹</td>
<td>Generic</td>
<td>Generic Layer 1</td>
<td>—</td>
<td>• NM-16A</td>
</tr>
<tr>
<td></td>
<td>connector</td>
<td></td>
<td></td>
<td>• NM-32A</td>
</tr>
<tr>
<td>generic-port¹</td>
<td>Generic</td>
<td>Generic Layer 1</td>
<td>Generic Layer 2</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>connector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>voiceEMDefault²</td>
<td>RJ45</td>
<td>Generic Layer 1</td>
<td>Generic Layer 2</td>
<td>• VIC-EM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• VIC-4VP-FXS-DID</td>
</tr>
</tbody>
</table>

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 22-25 lists templates that support modules for the Cisco Catalyst 3400 device group.

#### Module Group

These templates are defined in the ciscocatalyst3400spec module group.

### Table 22-25  Module Templates—Ethernet (Cisco Catalyst 3400)

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>cisco-3400-MultiTechnologies ModuleDefault</td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td>Ethernet</td>
<td>3400 fixed modules</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Not recommended for 10/100/1000 Ethernet ports. Instead, use cisco-3400-ethernetDefault-RJ45-or-Fiber. (See CSCTed93353 in CDETs.)</td>
</tr>
<tr>
<td>cisco-3400-ethernetDefault-RJ45-or-Fiber</td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td>• Ethernet</td>
<td>3400 fixed modules</td>
</tr>
<tr>
<td></td>
<td>• Fiber</td>
<td></td>
<td>• FastEthernet</td>
<td></td>
</tr>
<tr>
<td></td>
<td>optic</td>
<td></td>
<td>• GigaEthernet</td>
<td></td>
</tr>
</tbody>
</table>

### Ethernet (Cisco Catalyst)

**Note**  For modules in the Cisco Catalyst 3400 device group, see Ethernet (Cisco Catalyst 3400).

Table 22-26 lists templates that support modules for Cisco Catalyst devices.

#### Module Group

These templates are defined in the cisco-catalyst-spec module group.
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 22-24
- Supported Interface Types, page 22-25
- Event Templates Functional Summary, page 22-25
- Event Templates Input Summary—Required and Optional Input, page 22-27

For information about specific template types, see the following sections:

- Event Identification Templates, page 22-29
- Unique ID Templates—for Traps Only, page 22-31
- Event Subtype Templates, page 22-32
- Entity ID Templates—for Traps Only, page 22-34
- Entity Key Templates, page 22-35
- Prime Network Event Templates, page 22-37

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.

---

**Table 22-26 Module Templates—Ethernet (Cisco Catalyst)**

<table>
<thead>
<tr>
<th>Template Name</th>
<th>Layer 0</th>
<th>Layer 1</th>
<th>Layer 2</th>
<th>Example Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>EthernetDefault</td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td>Ethernet</td>
<td>• WS-C3560G-24TS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• WS-SUP720-3BXL</td>
</tr>
<tr>
<td>FastEthernetDefault</td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td>FastEthernet</td>
<td>• WS-X4148-RJ</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• ws-c2924-xl-v</td>
</tr>
<tr>
<td>GigaEthernetDefault</td>
<td>RJ45</td>
<td>EthernetCSMA/CD</td>
<td>GigaEthernet</td>
<td>• WS-F6K-MSFC2A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• OSM-2+4GE-WAN+</td>
</tr>
<tr>
<td>GigaEthernetOnCopper</td>
<td>UTP</td>
<td>EthernetCSMA/CD</td>
<td>GigaEthernet</td>
<td>Catalyst 6500 Supervisor Module 720 base board</td>
</tr>
<tr>
<td>giga-ethernet</td>
<td>Fiber optic</td>
<td>EthernetCSMA/CD</td>
<td>GigaEthernet</td>
<td>• cat6k-wsx-6066-slb-apc</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• wsx6ksup1a2ge</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>• wsx6ksup22ge</td>
</tr>
</tbody>
</table>
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 22-27 lists event templates (by type) and explains what each template does.

<table>
<thead>
<tr>
<th>Template</th>
<th>Performs This Function for a...</th>
<th>Syslog</th>
<th>Trap</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Event Identification Templates</strong>—Mandatory. Use one.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>snmp-trap-identification</td>
<td></td>
<td></td>
<td>Extracts information that identifies an event</td>
</tr>
<tr>
<td>syslog-identification</td>
<td>Extracts:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Information that identifies an event</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Event subtypes</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Entity ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Unique ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>snmp-trap-subtype-from-oid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>snmp-trap-subtype-from-trapoid</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>snmp-trap-subtype-from-value</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Event Subtype Templates**—Optional. Use one if there are subtypes for the event.

- Extracts subtype information from the event
- Maps the subtype values to the event subtypes defined in Prime Network
### Table 22-27  Event Template Functions (continued)

<table>
<thead>
<tr>
<th>Template</th>
<th>Performs This Function for a...</th>
</tr>
</thead>
<tbody>
<tr>
<td>syslog-subtype-from-expression</td>
<td>Maps the subtype values—extracted by the syslog-identification template—to the event subtypes defined in Prime Network</td>
</tr>
<tr>
<td>snmp-trap-identifier-from-oid</td>
<td>—</td>
</tr>
<tr>
<td>snmp-trap-identifier-from-value</td>
<td>—</td>
</tr>
<tr>
<td>snmp-trap-entity-from-oid</td>
<td>—</td>
</tr>
<tr>
<td>snmp-trap-entity-from-value</td>
<td>—</td>
</tr>
<tr>
<td>create-managedelement-key</td>
<td>• Creates the device component key</td>
</tr>
<tr>
<td></td>
<td>• Associates the event with the Managed Element device component</td>
</tr>
<tr>
<td>create-interface-key-from-ifindex</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>Associations the event with the appropriate interface layer; see Supported Interface Types, page 22-25</td>
</tr>
<tr>
<td>create-interface-key-from-ifname</td>
<td>• Creates the interface device component key from the ifname, using the entity ID that was extracted by:</td>
</tr>
<tr>
<td></td>
<td>• syslog-identification template for a syslog</td>
</tr>
<tr>
<td></td>
<td>• an entity ID template (snmp-trap-entity-from-oid or snmp-trap-entity-from-value) for a trap</td>
</tr>
<tr>
<td></td>
<td>• Associates the event with the appropriate interface layer; see Supported Interface Types, page 22-25</td>
</tr>
</tbody>
</table>

**Note** This template is more frequently used with syslogs than with traps.
Table 22-27  Event Template Functions (continued)

<table>
<thead>
<tr>
<th>Template</th>
<th>Performs This Function for a...</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Syslog</td>
</tr>
<tr>
<td>Prime Network Event Templates—Mandatory. Use one.</td>
<td></td>
</tr>
<tr>
<td>create-ana-trap-event</td>
<td>Creates both of the following:</td>
</tr>
<tr>
<td></td>
<td>• A unique location for the event,</td>
</tr>
<tr>
<td></td>
<td>based on the entity ID (device</td>
</tr>
<tr>
<td></td>
<td>component key) or the unique ID</td>
</tr>
<tr>
<td></td>
<td>(extracted by an entity ID or a</td>
</tr>
<tr>
<td></td>
<td>unique ID template).</td>
</tr>
<tr>
<td></td>
<td>• Prime Network event, using</td>
</tr>
<tr>
<td></td>
<td>subtypes extracted by an event</td>
</tr>
<tr>
<td></td>
<td>subtype template:</td>
</tr>
<tr>
<td></td>
<td>– snmp-trap-subtype-from-oid</td>
</tr>
<tr>
<td></td>
<td>– snmp-trap-subtype-from-trapoid</td>
</tr>
<tr>
<td></td>
<td>– snmp-trap-subtype-from-value</td>
</tr>
<tr>
<td>create-ana-syslog-event</td>
<td>Creates both of the following:</td>
</tr>
<tr>
<td></td>
<td>• A unique location for the event,</td>
</tr>
<tr>
<td></td>
<td>based on entity ID (device</td>
</tr>
<tr>
<td></td>
<td>component key) or unique ID</td>
</tr>
<tr>
<td></td>
<td>(extracted by the syslog-identification template).</td>
</tr>
<tr>
<td></td>
<td>• Prime Network event, using</td>
</tr>
<tr>
<td></td>
<td>subtypes extracted by the syslog-identification template</td>
</tr>
</tbody>
</table>

For the input required for each template, see Event Templates Input Summary—Required and Optional Input, page 22-27.

**Event Templates Input Summary—Required and Optional Input**

Table 22-28 summarizes event templates and the mandatory and optional input arguments for them.

Table 22-28  Event Template Variables

<table>
<thead>
<tr>
<th>Templates</th>
<th>Variable Name</th>
<th>Variable Type</th>
<th>Variable Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Identification Templates</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>syslog-identification</td>
<td>expression</td>
<td>Mandatory</td>
<td>Regular expression to match against the incoming syslog message.</td>
</tr>
<tr>
<td></td>
<td>testmessage</td>
<td>Optional</td>
<td>String that is an example of the actual syslog message.</td>
</tr>
<tr>
<td></td>
<td>oid</td>
<td>Mandatory</td>
<td>SNMP trap OID.</td>
</tr>
</tbody>
</table>
### Table 22-28 Event Template Variables (continued)

<table>
<thead>
<tr>
<th>Templates</th>
<th>Variable Name</th>
<th>Variable Type</th>
<th>Variable Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unique ID Templates</strong></td>
<td>snmp-trap-identifier-from-oid</td>
<td>inOID</td>
<td>Mandatory Partial string that uniquely identifies the required OID in the varbind list.</td>
</tr>
<tr>
<td></td>
<td>index</td>
<td>Optional</td>
<td>Used to create a unique location ID for the event when association to the exact entity is not possible. Default value is 1.</td>
</tr>
<tr>
<td></td>
<td>snmp-trap-identifier-from-value</td>
<td>inOID</td>
<td>Mandatory Partial string that uniquely identifies the required OID in the varbind list.</td>
</tr>
<tr>
<td><strong>Event Subtype Templates</strong></td>
<td>snmp-trap-subtype-from-oid</td>
<td>inOID</td>
<td>Mandatory Partial string that uniquely identifies the required OID in the varbind list.</td>
</tr>
<tr>
<td></td>
<td>replacing-rules</td>
<td></td>
<td>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 <code>vcb event add</code> command. To view event subtypes, use the <code>vcb event view</code> command.</td>
</tr>
<tr>
<td></td>
<td>index</td>
<td>Optional</td>
<td>Used to create a unique location ID for the event when association to the exact entity is not possible. Default value is 1.</td>
</tr>
<tr>
<td></td>
<td>snmp-trap-subtype-from-trapoid</td>
<td>replacing-rules</td>
<td>Mandatory Defines the mapping between event subtype and value in the trap that indicates the subtype.</td>
</tr>
<tr>
<td></td>
<td>snmp-trap-subtype-from-value</td>
<td>inOID</td>
<td>Mandatory Partial string that uniquely identifies the required OID in the varbind list.</td>
</tr>
<tr>
<td></td>
<td>replacing-rules</td>
<td></td>
<td>Defines the mapping between event subtype and value in the trap that indicates the subtype.</td>
</tr>
<tr>
<td></td>
<td>syslog-subtype-from-expression</td>
<td>replacing-rules</td>
<td>Mandatory Defines the mapping between event subtype and value in the trap that indicates the subtype.</td>
</tr>
<tr>
<td><strong>Entity ID Templates</strong></td>
<td>snmp-trap-entity-from-oid</td>
<td>inOID</td>
<td>Mandatory Partial string that uniquely identifies the required OID in the varbind list.</td>
</tr>
<tr>
<td></td>
<td>index</td>
<td>Optional</td>
<td>Used to create a unique location ID for the event when association to the exact entity is not possible. Default value is 1.</td>
</tr>
<tr>
<td></td>
<td>snmp-trap-entity-from-value</td>
<td>inOID</td>
<td>Mandatory Partial string that uniquely identifies the required OID in the varbind list.</td>
</tr>
<tr>
<td><strong>Entity Key Templates</strong></td>
<td>create-interface-key-from-ifindex</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>create-interface-key-from-ifname</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td></td>
<td>create-managedelement-key</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
Event Identification Templates

Event identification templates are mandatory. You must use one of these templates:

- `snmp-trap-identification`, page 22-29
- `syslog-identification`, page 22-30

**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 22-29 describes how to format the OID for different traps.

<table>
<thead>
<tr>
<th>If the OID...</th>
<th>Format Input Like This...</th>
<th>Example Input String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is for a V1 trap</td>
<td>Supply the enterprise OID appended by 0 and then by the specific type.</td>
<td>—</td>
</tr>
<tr>
<td>Contains subtype information as these do:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>- mplsLdpLibLspUp - 1.3.6.1.4.1.9.10.65.2.0.5</td>
<td>Remove subtype information from the input string.</td>
<td>1.3.6.1.4.1.9.10.65.2.0</td>
</tr>
<tr>
<td>- mplsLdpLibLspDown - 1.3.6.1.4.1.9.10.65.2.0.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Event Templates

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
```

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\$+:clock (\$+).*
```

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:

```
mplsLdpPathVectorLimitMismatch - 1.3.6.1.4.1.9.10.65.2.0.2
```

Supply the entire OID.

```
demandNbrLayer2Change - 1.3.6.1.4.1.9.9.26.2.0.3
```

Supply the entire OID.

```
%C6KENV-4-CLOCK%%subtypekey%%: clock %%uniqueid%%
```

Table 22-29  Template Input—snmp-trap-identification

<table>
<thead>
<tr>
<th>If the OID...</th>
<th>Format Input Like This...</th>
<th>Example Input String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is an informational trap as this one is: mplsLdpPathVectorLimitMismatch - 1.3.6.1.4.1.9.10.65.2.0.2</td>
<td>Supply the entire OID.</td>
<td>1.3.6.1.4.1.9.10.65.2.0.2</td>
</tr>
<tr>
<td>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</td>
<td>Supply the entire OID.</td>
<td>1.3.6.1.4.1.9.9.26.2.0.3</td>
</tr>
<tr>
<td>The subtype for this trap is a varbind: isdnLapdOperStatus.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 22-29  Template Input—snmp-trap-identification

<table>
<thead>
<tr>
<th>If the OID...</th>
<th>Format Input Like This...</th>
<th>Example Input String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is an informational trap as this one is: mplsLdpPathVectorLimitMismatch - 1.3.6.1.4.1.9.10.65.2.0.2</td>
<td>Supply the entire OID.</td>
<td>1.3.6.1.4.1.9.10.65.2.0.2</td>
</tr>
<tr>
<td>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</td>
<td>Supply the entire OID.</td>
<td>1.3.6.1.4.1.9.9.26.2.0.3</td>
</tr>
<tr>
<td>The subtype for this trap is a varbind: isdnLapdOperStatus.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 22-29  Template Input—snmp-trap-identification

<table>
<thead>
<tr>
<th>If the OID...</th>
<th>Format Input Like This...</th>
<th>Example Input String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is an informational trap as this one is: mplsLdpPathVectorLimitMismatch - 1.3.6.1.4.1.9.10.65.2.0.2</td>
<td>Supply the entire OID.</td>
<td>1.3.6.1.4.1.9.10.65.2.0.2</td>
</tr>
<tr>
<td>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</td>
<td>Supply the entire OID.</td>
<td>1.3.6.1.4.1.9.9.26.2.0.3</td>
</tr>
<tr>
<td>The subtype for this trap is a varbind: isdnLapdOperStatus.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
%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 22-29.

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 22-31
- snmp-trap-identifier-from-value, page 22-32

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 22-30.)
- 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 22-32
- **snmp-trap-subtype-from-trapoid**, page 22-33
- **snmp-trap-subtype-from-value**, page 22-33

When subtypes exist for a syslog, use this template:

- **syslog-subtype-from-expression**, page 22-34.

**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

-\texttt{-snmp\_trap\_subtype\_from\_oid\_inOID} OID
-\texttt{-snmp\_trap\_subtype\_from\_oid\_index} indexValue
-\texttt{-snmp\_trap\_subtype\_from\_oid\_replacing\_rules} value-subtype, value-subtype, value-subtype

Note
The hyphen between value and subtype is required.

\texttt{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

-\texttt{-snmp\_trap\_subtype\_from\_trapoid\_replacing\_rules} value-subtype, value-subtype, value-subtype

\texttt{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 22-29.

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 22-35
- `snmp-trap-entity-from-value`, page 22-35
snmp-trap-entity-from-oid

The rules in this template extract the entity ID from 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`

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 22-30 and Entity ID Templates—for Traps Only, page 22-34.

Select one of the following templates:

- create-interface-key-from-ifindex, page 22-36
- create-interface-key-from-ifname, page 22-36
- create-managedelement-key, page 22-36
- create-efp-key-from-ifname-serviceid, page 22-36
- create-logical-container-key, page 22-36
- create-moduleDC-key-given-entPhysicalIndex, page 22-36
- create-moduleDC-with-slotSubslot-value-key, page 22-37
- create-pw-interface-key-from-tunnelindex, page 22-37
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.

\[\text{Note}\]
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,Cl
ockService. 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.
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-REMCARD: 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](Image)
  
  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](Image)
  
  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 eventparsingrules Commands`, page 20-21.
PART 6

External Launch

This part contains the following chapters:

- Chapter 23, “Adding External Launch Points”
- Chapter 24, “Sample BQL Scripts to Launch External Applications”
Adding External Launch Points

This chapter describes how to add launch points from Prime Network to external applications. Topics include:

- About Adding External Launch Points, page 23-1
- Adding an External Launch Point, page 23-2
- Accessing the External Launch Point in Prime Network, page 23-7
- Deleting an External Launch Point, page 23-8

About Adding External Launch Points

If you have Configurator or higher privileges, Prime Network allows you to add launch points in Prime Network Vision or Prime Network Events to launch an external application, command, or URL. You can add launch points to any IMO, including network elements, links, tickets, and events. These launch points appear as additional right-click menu options. When the launch point is clicked, it invokes a script or batch file that you write to define the application or command to be launched, or accesses a specified URL.

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 Scripts to Clients, page 23-7, to enable the automatic download of the scripts to the clients.
- A shared location (like Cisco's babab).
Adding an External Launch Point

Adding an external launch point involves the following steps:

1. Identifying the IMO Context and Properties, page 23-2
2. Creating BQL Scripts, page 23-3
3. Running BQL Scripts, page 23-7
4. Enabling Automatic Download of Scripts to Clients, page 23-7

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:

```java
com.sheer.imo.IManagedElement
{[ManagedElement (Key=ana7609-1)]}
```

Example 2: Identifying IPortConnector in the Tree pane:

```java
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:

```java
{[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

```java
key=DeviceName, value=ana7609-1
key=IP, value=172.20.68.72
```

Example 2: IPortConnector

```java
key=Location, value=5.GigabitEthernet5/1 - Missing Pluggable Port
key=PortAlias, value=GigabitEthernet5/1 - Missing Pluggable Port
```
Creating BQL Scripts

Read *Cisco Prime Network 3.8 Integration Developer Guide* for an understanding of Broadband Query Language (BQL), and as a prerequisite to understanding this section.

Table 23-1 describes the format and syntax that you must follow when writing the BQL set command to launch an external application:

**Table 23-1 BQL Command Format and Syntax**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
</table>
| ID       | OID    |       | 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 sub-type, like ISyslogAlarm)—Events and their sub-types.  
  - ITicket—Tickets.  
  - ITopologicalLink—Links. |
| Filter tags | String | Example:  
  - To filter by vendor name: `<Vendor type="String">Cisco</Vendor>`  
  - To filter by latest state: `<LatestState type="String">Down</LatestState>` | Filter tags allow you to set filters for the external launch commands. The following filter tags are supported for the supported IMOs:  
  - IManagedElement—ElementType, Vendor  
  - IEvent (or sub-type)—Description, State  
  - ITicket—LatestState  
  - ITopologicalLink—LinkType. |
Adding an External Launch Point

Adding an External Launch Point

ExternalBatchToExecute

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:
  `<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>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExternalBatchToExecute</td>
<td>String</td>
<td>Examples:</td>
<td>Defines the details of the executable steps or batch file to be invoked. You:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Specifying a batch file located on the client machine:</td>
<td>- Must specify the parameters as an array when using a script.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;ExternalBatchToExecute type=&quot;String&quot;&gt;C:\runthis.bat&lt;/ExternalBatchToExecute&gt;</code></td>
<td>- Cannot pass any parameters in the URL. You must use batch files or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Specifying a batch file located on the gateway machine:</td>
<td>LineToExecute to pass the parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;ExternalBatchToExecute type=&quot;String&quot;&gt;foldername/runthis.bat&lt;/ExternalBatchToExecute&gt;</code></td>
<td>- Can include more commands as part of a batch file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Specifying a URL:</td>
<td>- Can include an environment variable when invoking the batch file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;ExternalBatchToExecute type=&quot;String&quot;&gt;http:\www.cisco.com&lt;/ExternalBatchToExecute&gt;</code></td>
<td>To define a string that represents the URL or script which is executed</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Passing an environment variable:</td>
<td>along with its parameters, use the LineToExecute property.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;ExternalBatchToExecute type=&quot;String&quot;&gt;%java_home%\bin\javadoc&lt;/ExternalBatchToExecute&gt;</code></td>
<td></td>
</tr>
<tr>
<td>LineToExecute</td>
<td>String</td>
<td>Example:</td>
<td>Defines a string that represents the URL or script which is executed along</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Launching a URL with the Gateway IP, network element IP, and logged-in</td>
<td>with its parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>user name parameters, using / as delimiter:</td>
<td>The syntax includes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;LineToExecute type=&quot;String&quot;&gt;http://url.example.com/$GW$/&lt;/LineToExecute&gt;</code></td>
<td>- $—Used when you specify an IMO property definition.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- ~—Used when you specify a Soft Property definition.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- $number$—The parameter whose index is a number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- $USERNAME$—The logged-in username.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- $GW$—The Prime Network gateway IP address.</td>
</tr>
</tbody>
</table>

Note: The & character should be written as &amp; when used inside a BQL command.
Table 23-1  BQL Command Format and Syntax (continued)

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MenuCaption</td>
<td>String</td>
<td>Example:</td>
<td>The wording of the right-click menu option to be used for the external launch.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;MenuCaption type=&quot;String&quot;&gt;ping&lt;/MenuCaption&gt;</td>
<td></td>
</tr>
<tr>
<td>MenuPath</td>
<td>String</td>
<td>Example:</td>
<td>The menu path to be followed to get to the external launch point. You can have a sub-menu separated by a forward slash (/).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;MenuPath type=&quot;String&quot;&gt;external launch demo&lt;/MenuPath&gt;</td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td>IMObjects</td>
<td>&lt;Parameters type=&quot;IMObjects_Array&quot;&gt;</td>
<td>All external launch parameters must be passed in an array. See management.IExternalLaunchParameter, page 23-5.</td>
</tr>
<tr>
<td></td>
<td>Array</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Role</td>
<td>String</td>
<td>Example:</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;Role type=&quot;string&quot;&gt;CONFIGURATOR&lt;/Role&gt;</td>
<td></td>
</tr>
<tr>
<td>Selection Limit</td>
<td>Int</td>
<td>Example:</td>
<td>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 1 item is selected, or if 3 items are selected. It will only appear if 2 items are selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;SelectionLimit type=&quot;int&quot;&gt;2&lt;/SelectionLimit&gt;</td>
<td></td>
</tr>
</tbody>
</table>

**management.IExternalLaunchParameter**

<table>
<thead>
<tr>
<th>ID</th>
<th>OID</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>{{{ExternalLaunchParameter(Index=1)}}}</td>
<td>Index value defines the order in which the parameters will be passed to the script.</td>
</tr>
<tr>
<td>IMOType</td>
<td>String</td>
<td>Example 1:</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;IMOType type=&quot;String&quot;&gt;com.sheer.imo.IPortConnector&lt;/IMOType&gt;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example 2:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&lt;IMOType type=&quot;String&quot;&gt;com.sheer.imo.newalarm.ITicket&lt;/IMOType&gt;</td>
<td></td>
</tr>
</tbody>
</table>
### Adding an External Launch Point

#### Defining Soft Property Parameters

You can use a soft property as a parameter in an external launch BQL command.

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:

```xml
<SoftProperty type="boolean">true</SoftProperty>
```

If you do not add this entry, by default the value is assumed to be false.

---

**Table 23-1  BQL Command Format and Syntax (continued)**

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertyName</td>
<td>String</td>
<td>Example 1:</td>
<td>Defines the IMO key name within the IMO context type. This value is used by the script while invoking the application.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;PropertyName type=&quot;String&quot;&gt;PortAlias&lt;/PropertyName&gt;</code></td>
<td>You can also have any constant value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example 2:</td>
<td>If the property name includes another IMO, you can retrieve the properties of this IMO.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;PropertyName type=&quot;String&quot;&gt;LatestState&lt;/PropertyName&gt;</code></td>
<td>For details about adding Soft Property parameters, see Defining Soft Property Parameters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example 3 (retrieving properties of a contained IMO):</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;PropertyName type=&quot;String&quot;&gt;ObjectId.Key&lt;/PropertyName&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example 4 (using a business tag key as a parameter):</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;PropertyName type=&quot;String&quot;&gt;@IBusinessObject.EKey&lt;/PropertyName&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Defines the IMO key name within the IMO context type. This value is used by the script while invoking the application.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>This value is used by the script while invoking the application.</td>
<td>You can also have any constant value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If the property name includes another IMO, you can retrieve the properties of this IMO.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>For details about adding Soft Property parameters, see Defining Soft Property Parameters.</td>
<td></td>
</tr>
<tr>
<td>RunSingleCommand</td>
<td>Boolean</td>
<td>Example:</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;RunSingleCommand type=&quot;boolean&quot;&gt;true&lt;/RunSingleCommand&gt;</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>The script will be called once, with the required properties of all selected IMOs passed at the same time.</td>
<td></td>
</tr>
<tr>
<td>ReplaceNullWith</td>
<td>String</td>
<td>Example:</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><code>&lt;ReplaceNullWith type=&quot;String&quot;&gt;N/A&lt;/ReplaceNullWith&gt;</code></td>
<td>In the example, the null value will be replaced by N/A. See Defining Soft Property Parameters for more information on the usage of this property.</td>
</tr>
</tbody>
</table>

---

**Defining Soft Property Parameters**

You can use a soft property as a parameter in an external launch BQL command.

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:

```xml
<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. This is because there is no means to discover whether the IMO on which the command is run has the soft property defined on it.

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 3.8 Integration Developer Guide* to understand how to run a BQL script. This guide is located in the Prime Network Technology Center on the Cisco Developer Network (CDN), under the Integration tab.

---

**Enabling Automatic Download of Scripts to Clients**

In Prime Network 3.8, 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.

---

**Accessing the External Launch Point in Prime Network**

To launch an external application:

**Step 1** Log into Prime Network Vision.

**Step 2** Navigate to the IMO context(s) where you added the external launch point.
Deleting an External Launch Point

You can delete an external application launch point using the BQL **Delete** command.

---

**Note**

See the *Cisco Prime Network 3.8 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
<?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 to Launch External Applications

This chapter provides example scripts for launching an external application using BQL.

Note See BQL Command Format and Syntax for details on the syntax used in the commands.

- **Example 1**—Run a single script on the selected tickets with their latest state and last modification time as parameters.
- **Example 2**—Run 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.
- **Example 3**—Include a soft property parameter in a BQL command for an external launch point on a network element.
- **Example 4**—Use a Batch File to Invoke an Application.
- **Example 5**—Execute a script C:/script.batch with parameters PortAlias and the logged-in user name.
- **Example 6**—Launch a URL with parameters values Gateway IP, the Network Element IP and the logged-in user name, using / as delimiter.
- **Example 7**—Launch a URL with parameters a, b and c whose values are the Soft property showClock, a constant string "constant" and the logged-in user name.
- **Example 8**—Launch 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 user name, using the existing Parameters element.
- **Example 9**—Using an Environment Variable to Invoke an Application.
- **Example 10**—Using Multiple IMO Contexts to Launch an Application.

For details about the format and the syntax of the BQL script, see Creating BQL Scripts, page 23-3.

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**

Running 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

Running 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

Including a soft property parameter in a BQL command for an external launch point on a network element:

```xml
<command name="Set">
  <param name="imo">
    <value>
      <management.IExternalLaunch>
        <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IMangedElement)(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.IMangedElement</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.IMangedElement</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.IMangedElement</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

Using a Batch File to Invoke an Application

```xml
<command name="Set">
  <param name="imo">
    <value>
      <management.IExternalLaunch>
      </value>
  </param>
  <param name="replace">
    <value>true</value>
  </param>
</command>
```
Example 5

Execute a script C:/script.batch with parameters PortAlias and the logged-in user name.

```xml
<?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

Launch a URL with parameters values Gateway IP, the NE IP and the logged-in user name, using / (forward slash) as delimiter.

```xml
<?xml version="1.0" encoding="UTF-8"?
<command name="Set">
  <param name="imo">
    <value>
      <management.IExternalLaunch>
        <ID type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPDealHost) (Name=example)]}</ID>
        <ElementType type="String">IOU</ElementType>
        <Vendor type="String">Cisco</Vendor>
      </management.IExternalLaunch>
    </value>
  </param>
  <param name="replace">
    <value>true</value>
  </param>
</command>
```
Example 7

Launch a URL with parameters a, b, and c whose values are the Soft property showClock, a constant string constant, and the logged-in user name.

```xml
<?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.imo.IManagedElement.showClock~&b=constant&amp;c=$USERNAME$</LineToExecute>
      </management.IExternalLaunch>
    </value>
  </param>
  <param name="replace">
    <value>true</value>
  </param>
</command>
```

Example 8

Launch 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 user name, using the existing Parameters element.

```xml
<?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=$1$&b=$2$&c=$3$&d=$USERNAME$</LineToExecute>
        <Parameters type="IMObjects_Array">
          <management.IExternalLaunchParameter>
            <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</ID>
            <IMOType type="String">com.sheer.imo.IManagedElement</IMOType>
          </management.IExternalLaunchParameter>
        </Parameters>
      </management.IExternalLaunch>
    </value>
  </param>
  <param name="replace">
    <value>true</value>
  </param>
</command>
```
Example 9

Using an Environment Variable to Invoke an Application

```xml
<command name="Set">
  <param name="imo">
    <value>
      <management.IExternalLaunch>
        <ID type="Oid">{[ExternalLaunchParameter(Index=1)]}</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">com.sheer.imo.IManagedElement</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

Using 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
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
type="Oid">{[ExternalLaunch(ContextImoType=com.sheer.imo.IPortConnecto
        <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.IMangedElement</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">any_const_parameter</ IMOType>
          </management.IExternalLaunchParameter>
        </Parameters>
      </management.IExternalLaunch>
    </value>
  </param>
  <param name="replace">
    <value>true</value>
  </param>
</command>
PART 7

Drools Rules

This part contains the following chapters:

- Chapter 25, “Drools Rules Management”
- Chapter 26, “Drools Rules Examples”
Drools Rules Management

This chapter describes the Drools Rules engine and its usage in Cisco Prime Network. Topics include:

- About the Drools Rules Engine, page 25-1
- Drools Rules Definitions in Prime Network, page 25-3
- Displaying Existing Drools Rules, page 25-4
- Upgrading and Validating Drools Rules Files, page 25-4
- About Drools Rules Migration, page 25-5

Note
Contact a developer support representative if you want to add a new Drools Rule to manage and troubleshoot the fault in your network. For a list of class associations to help you select the condition to be included in the rule, see the Prime Network 3.8 IMO documentation available at http://developer.cisco.com/web/prime-network/home.

Additional information about Drools Rules is available on the Cisco Developer Network (CDN).

About the Drools Rules Engine

The Drools Rules engine is a general-purpose expert system generator that combines rule-based techniques and object-oriented programming. Drools implements and extends the Rete algorithm, and uses the rule based approach to implement an expert system.

Drools is based on an object-oriented paradigm and uses user-defined rules to perform pattern matching on different conditions. It also provides a customizable mechanism to add decision support and data-flow control functions to business applications.
The Drools syntax is XML-based with embedded Java code, and is organized into source files (known as a rule files), which are plain ASCII files. Drools uses objects as marked by patterns and rules that invoke certain actions.

Additional notes:
- Drools objects are Java objects and can be represented by instances of Java classes or XML schemas.
- A pattern is a coded expression (program) that manipulates one or more objects to form a pattern to make, adapt, or fashion behavior according to designed logic.
- Working memory is where Drools stores all objects that it is handling.
- Actions are operations that might change the working memory.
- A rule can perform many types of actions, such as, executing a method on one of the objects.
- Drools stores in the agenda the list of rules to be executed.

For more information about Drools Rules see www.jboss.org/drools/

Drools Rules and Prime Network

Prime Network uses Drools version 2.1. The Drools Rules engine enables you to extend the Prime Network alarm correlation mechanism with user-defined rules and business logic. It can be used to enable business rules to be applied on event updates.

Drools Rules can be triggered on tickets, network events, and non-network events.

Rules are applied after new ticket is initiated or a new event (network or non-network event) is initiated in Prime Network. Non-actionable events are not supported by the Drools Rules engine.

The Drools Rules engine is fully integrated within the Prime Network gateway and does not require any synchronization or maintenance.

A primary advantage of the Prime Network Drools Rules engine is that it lets you act upon certain events or tickets, and fully customize the action to the case. Here are some examples:
- Notify a network operator by e-mail when various network faults occur.
- Notify a network administrator using SMS in the case of a major network fault.
- Notify the Prime Network system administrator when system events occur, such as running out of disk space. In addition, run a disk cleaning script.
- Notify the security administrator when security events occur.

Some of these examples, such as sending SMS or e-mail, require interaction with third party systems not provided with Prime Network.

Drools rules can be added to Prime Network by creating new rule definitions in the ~/Main/data/post.drl file located on the Prime Network gateway server. The rules are written in XML format with embedded Java code. There is no need to compile the rules.

Each rule contains a set of conditions and is executed only if the condition is true. Drools rules contain an XML entry similar to the following, which specifies whether or not they are enabled:

```xml
<java:condition>true</java:condition>
```
All enabled rules are loaded by the Prime Network gateway and are triggered on the relevant Prime Network objects.

Consider the following points before you build and test the rules:

- Error and processing messages are written to ~/Main/logs/11.out on the Prime Network gateway server.
- We recommend that you use this capability with some caution, as the more rules that are defined, the greater the performance impact on the system.

## Enabling and Disabling Drools Rules

In Prime Network, the Drools Rules engine is disabled by default. If Drools Rules are to be used, the engine should be explicitly enabled.

To enable the Drools Rules engine:

**Step 1**

From ~/Main, enter the following commands:

- ./runRegTool.sh -gs 127.0.0.1 set 0.0.0.0 site/plugin/droolsplugin/enable true
- ./runRegTool.sh -gs 127.0.0.1 add 0.0.0.0 site/plugin/AlarmPlugin/NotificationHandlers/com.sheer.metromission.plugin.alarm.DroolsNotificationHandler

**Step 2**

Restart AVM11 to apply this change.

If no rules are in use, we recommend that you disable the Drools Rules engine.

To disable the Drools Rules engine, do the following:

**Step 1**

From ~/Main, enter the following commands:

- ./runRegTool.sh -gs 127.0.0.1 set 0.0.0.0 site/plugin/droolsplugin/enable false
- ./runRegTool.sh -gs 127.0.0.1 remove 0.0.0.0 site/plugin/AlarmPlugin/NotificationHandlers/com.sheer.metromission.plugin.alarm.DroolsNotificationHandler

**Step 2**

Restart AVM11 to apply this change.

## Drools Rules Definitions in Prime Network

Drools supports ticket and event IMO manipulation for each ticket/event processed by the Prime Network gateway. Prime Network maintains the following Drools processing instance (context) for the rules file:

- Postcorrelation context—Defined in the post.drl rule file. This rule is executed after processing in the Prime Network gateway.

The rule file is located under ~/Main/data in the Prime Network gateway server.
The following Drools Rules parameters are stored in the registry file mmvm.xml under the eventmanagement key:

- ContextId—The Drools context name.
- ruleFilePath—The name of the respective rule file. The specified rule (the filename under mmvm.xml).

The mmvm.xml file is located under ~/Main/registry/ConfigurationFiles/127.0.0.1.

### Modifying a Rule

You can modify a rule. After you edit a rule, you need to reload the rule.

To reload the rule, run the following BQL command:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<command name="ReloadRules">
  <param name="contextId">
    <value>POST</value>
  </param>
</command>
```

### Displaying Existing Drools Rules

You can check the supported Prime Network postcorrelation rules by using the following BQL command:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<command name="GetAllRules">
  <param name="contextId">
    <value>POST</value>
  </param>
</command>
```

### Upgrading and Validating Drools Rules Files

To upgrade a rule file:

1. **Step 1** Make a copy of the post.drl file, and edit it.
2. **Step 2** Copy the updated file (under a temporary name) to the gateway (directory ~sheer/Main/data).

   **Note** If the rule file is edited on a PC, make sure that the text format is compliant to the UNIX version that runs on the gateway. If necessary, use a utility such as DOS2UNIX for the conversion.

3. **Step 3** Check the validity of the new file by running the following BQL command:
<?xml version="1.0" encoding="UTF-8"?>
<command name="ValidateRules">
  <param name="fileName">
    <value>rule-file-name-full-path</value>
  </param>
</command>

For example:

<?xml version="1.0" encoding="UTF-8"?>
<command name="ValidateRules">
  <param name="fileName">
    <value>/export/home/network38/Main/data/post.drl</value>
  </param>
</command>

**Step 4** Once the rule file has been validated:

a. Copy the new rule file over the existing rule file.

b. Reload rules files by running the following BQL command:

<?xml version="1.0" encoding="UTF-8"?>
<command name="ReloadRules">
  <param name="contextId">
    <value>POST</value>
  </param>
</command>

**About Drools Rules Migration**

If you upgrade from Cisco ANA 3.7.3 to Prime Network 3.8, your existing post.drl file and customization remain intact and are available after upgrading.

If you upgrade from Cisco ANA 3.6.x, review the Drools Rules changes and migration guidelines in the *Cisco Active Network Abstraction 3.7 Customization User Guide*. 
Drools Rules Examples

This chapter provides examples to illustrate how Drools Rules can be used in Cisco Prime Network 3.8 to enable alarm forwarding using e-mails.

Topics include:

- Example 1: Generate E-mail Upon Link Down/Up Ticket, page 26-1
- Example 2: Generate E-mail for Critical Ticket, page 26-2

Example 1: Generate E-mail Upon Link Down/Up Ticket

This example is a of a Drools Rule that generates an e-mail if a LinkDown / LinkUp Ticket is raised.

```
<rule name="Severity critical Alert">
  <parameter identifier="networkEvent">
    <class>com.sheer.imo.newalarm.INetworkEvent</class>
  </parameter>
  <java:condition>networkEvent.getName() == 1</java:condition>
  <java:consequence>
    import javax.mail.*;
    import javax.mail.internet.*;
    import java.util.*;
    import com.sheer.system.os.interfaces.Logger;
    Logger.getInstance().fatal("\nCritical Ticket Rule Fired\n");
    String recipients[] = new String[2] {"<emailID1>", "<emailID2>"};
    String subject = networkEvent.getDescription();
    String message = "A Link Down/up Ticket has been raised. Please attend to the problem immediately";
    String from = "CiscoANA@company.com";
    boolean debug = false;
    // Set the host smtp address
    Properties props = new Properties();
    props.put("mail.smtp.host", "<smtp server name>");
    // create some properties and get the default Session
    Session session = Session.getDefaultInstance(props, null);
    session.setDebug(debug);
    // create a message
    Message msg = new MimeMessage(session);
    // set the from and to address
    InternetAddress addressFrom = new InternetAddress(from);
    msg.setFrom(addressFrom);
    InternetAddress[] addressTo = new InternetAddress[recipients.length];
    for (int i = 0; i < recipients.length; i++) {
      addressTo[i] = new InternetAddress(recipients[i]);
    }
```
Example 2: Generate E-mail for Critical Ticket

The following Drools Rule example generates an email if a Ticket with severity Critical has been raised.

Only new ticket notifications will be captured here—updates on this ticket will not be received. For example, if a ticket is initiated with Aggregated Severity MINOR and then due to correlation with a critical alarm the severity was changed to CRITICAL, it won't be captured in this rule.

```
<rule name="Severity critical Alert">
  <parameter identifier="ticket">
    <class>com.sheer.imo.newalarm.ITicket</class>
  </parameter>
  <java:condition>ticket.getAggregatedSeverityEnum() == 6</java:condition>
  <java:consequence>
    import javax.mail.*;
    import javax.mail.internet.*;
    import java.util.*;
    import com.sheer.system.os.interfaces.Logger;
    import com.sheer.imo.newalarm.ITicket;
    import com.sheer.imo.keys.INewAlarmOid;
    import com.sheer.client.common.util.DisplayNameFactory;
    Logger.getInstance().fatal("\nCritical Ticket Rule Fired\n");
    String recipients[] = new String[] {"<emailID1>", "<emailID2>"};
    String subject = ticket.getDescription();
    String message = "Ticket ID = " + ((INewAlarmOid)ticket.getObjectId()).getId() + "\n" +
    "Severity = " + ITicket.S_AGGREGATED_SEVERITY[ticket.getAggregatedSeverityEnum()] + "\n" + "Creation Time = " + ticket.getCreationTime() + "\n" + "Location = " + DisplayNameFactory.getTextForIOid(ticket.getSource());
    String from = "CiscoANA@company.com";
    boolean debug = false;
    //Set the host smtp address
    Properties props = new Properties();
    props.put("mail.smtp.host", "<smtp server name>");
    // create some properties and get the default Session
    Session session = Session.getDefaultInstance(props, null);
    session.setDebug(debug);
    // create a message
    Message msg = new MimeMessage(session);
    // set the from and to address
    InternetAddress addressFrom = new InternetAddress(from);
    msg.setFrom(addressFrom);
    InternetAddress[] addressTo = new InternetAddress[recipients.length];
    for (int i = 0; i < recipients.length; i++) {
      addressTo[i] = new InternetAddress(recipients[i]);
    }
    msg.setRecipients(Message.RecipientType.TO, addressTo);
    // Setting the Subject and Content Type
    msg.setSubject(subject);
    msg.setContent(message, "text/plain");
    Transport.send(msg);
  </java:consequence>
</rule>
```
Example 2: Generate E-mail for Critical Ticket

```java
Transport.send(msg);
</java:consequence>
</rule>

Note
To create this rule for a MAJOR ticket, change the <java:condition> section as follows:
<java:condition> ((ticket.getAggregatedSeverityEnum() == 6) || (ticket.getAggregatedSeverityEnum() == 5))</java:condition>

The Properties object in the above examples is used only the first time Session.getDefaultInstance(props, null) method is called, when a new Session object is created. Subsequent calls return the Session object that was created by the first call, and ignore the passed Properties object. Use the Session.getInstance(props, null) method to get a new Session object every time the method is called (for example, if you need to change the Mail SMTP Host).

The following table describes the properties used in the above examples.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rule name</td>
<td>The name of the rule.</td>
</tr>
<tr>
<td>parameter identifier</td>
<td>A definition of a parameter to use in this rule context. The parameter value will be any instance of the class defined within the &lt;class&gt;&lt;/class&gt; brackets.</td>
</tr>
<tr>
<td>java:condition</td>
<td>The condition that initiates the e-mail generation, based on the rule definitions. For Example 1, the condition is that a LinkDown/Up ticket is raised. For Example 2, the condition is that a ticket with Critical severity is raised.</td>
</tr>
<tr>
<td>java:consequence</td>
<td>The set of Java operations to run in case the previous conditions have been met. This section should include your drool rule operations.</td>
</tr>
<tr>
<td>Logger:getInstance</td>
<td>Helps to identify whether a rule was triggered. After you reload the rule, go to ~/Main/logs directory and enter tail -f 11.out. Check the log file to find out whether the rule was triggered.</td>
</tr>
<tr>
<td>String recipients</td>
<td>The e-mail IDs to which you want the system to send an e-mail.</td>
</tr>
<tr>
<td>String subject</td>
<td>The string to fetch the description for the alarm / ticket. In example 1, the string is networkEvent.getDescription().</td>
</tr>
<tr>
<td>String from</td>
<td>The e-mail ID to be displayed in the From field.</td>
</tr>
</tbody>
</table>
A

alarm thresholds
   about 2-2
   triggers 6-1

B

BeanShell commands
   about 16-1
   defining scripts 13-16
   SNMP examples 16-3
   Telnet examples 16-2

C

Command Builder
   about 1-3
   BeanShell commands, see BeanShell commands
   Command Builder window 13-2
   creating a command 13-7
      example 15-1
   defining a combo field 13-13
   defining script lines 13-14
   defining user input parameters 13-9
   deleting commands 13-30
   editing a command 13-26
   executing commands 13-18
   Macro Language, see Macro Language
   menu options 13-4
   previewing commands 13-18
   publishing commands 13-24
   reviewing command history 13-30
   running command on multiple NEs 12-1, 13-16, 13-17, 13-24, 13-26
   toolbar 13-4
   workflow to define command definition 13-6

conventions, in this guide xvi
   customization components 1-2
   custom syslogs
      adding support for 20-7

D

Deleting Modules 19-5
   deleting VCB registry customizations 21-3
   device type
      adding new 18-15
      deleting 18-16
      editing 18-15
      enabling support by creating U-VNE 18-1
      using CLI to manage 18-31
      viewing existing 18-14
   document
      conventions xvi
   Drools Rules engine
      about 25-1
      and Prime Network 25-2
      components and terminology 25-2
      definitions 25-3
      show rule 25-4
      upgrading rule files 25-4

E

   events
Index

deleting 20-12
dropping 20-11
enabling support for 20-1
moving to production 20-13
restoring a dropped event 20-11
testing 20-12
troubleshooting event customization 20-34

changing severity 20-10
external launch points
accessing in Prime Network 23-7
adding 23-1
deleting 23-8

G
genericUVNE template 22-1

I
importing VCB registry customizations 21-3

L
Launching External Applications
deleting 23-8
launching 23-7

log file
VCB 17-4

M
Macro Language
about scripts 14-1
built-in parameters 14-4
defining script lines 13-14
defining example of creating a command 15-1
formats for IP subnet parameters 14-4

properties 14-2
script example 14-9
script parameters 14-2
supported pragmas 14-4
understanding 14-1
user-defined parameters 14-3

Managed Element 2-3
modules
commands for managing 19-10
editing 19-4
enabling support for 19-1
evaluating module templates 19-9
using CLI to manage 19-9
viewing 19-3

modules, managing additional using the VCB module group
ciscocatalyst3400spec 22-14
cisco-catalyst-spec 22-14
ciscophysicsalspec2

module template, example module
16OC48-POS/DPT 22-16
1OC768-ITU/C 22-17
2DS3-SMB 22-19

3400 fixed modules 22-23
7600-ES+2TG 22-14, 22-15
7600-ES+3C 22-14, 22-15
76-ES+XT-2TG3CX 22-15
76-ES+XT-4TG3C 22-15
8FE-TX-RJ45 22-14, 22-15
cat6k-wsx-6066-slb-apc 22-23

Catalyst 6500 Supervisor Module 720 base board 22-23
ccevCpu2941 22-15
copper-12ds3 22-19
copper-6ds3 22-19
ESR-8E3/DS3 22-18
GSR-CSC 22-16
gsr-e-qoc48-sm-lr-sc 22-16
GSR-SFC12410 22-19
<table>
<thead>
<tr>
<th>Module Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>gsr-sfc16-oc192</td>
<td>22-20</td>
</tr>
<tr>
<td>GSR-SFC6</td>
<td>22-16</td>
</tr>
<tr>
<td>HWIC-4T</td>
<td>22-21</td>
</tr>
<tr>
<td>HWIC-4T1/E1</td>
<td>22-18</td>
</tr>
<tr>
<td>NM-16A</td>
<td>22-22</td>
</tr>
<tr>
<td>NM-1A-E3</td>
<td>22-19</td>
</tr>
<tr>
<td>NM-1CE1T1-PRI</td>
<td>22-19</td>
</tr>
<tr>
<td>NM-1T3/E3</td>
<td>22-19</td>
</tr>
<tr>
<td>NM-2W</td>
<td>22-21</td>
</tr>
<tr>
<td>NM-32A</td>
<td>22-22</td>
</tr>
<tr>
<td>NM-4T</td>
<td>22-19</td>
</tr>
<tr>
<td>OSM-1CHOC12/T3-SI</td>
<td>22-20</td>
</tr>
<tr>
<td>OSM-1OC48-POS-SI+</td>
<td>22-15</td>
</tr>
<tr>
<td>OSM-2+4GE-WAN+</td>
<td>22-15, 22-23</td>
</tr>
<tr>
<td>OSM-2OC12-POS-SI</td>
<td>22-15</td>
</tr>
<tr>
<td>OSM-2OC12-POS-SI+</td>
<td>22-15</td>
</tr>
<tr>
<td>OSM-4OC12-POS-SI+</td>
<td>22-15</td>
</tr>
<tr>
<td>OSM-4OC3-POS-SI+</td>
<td>22-15</td>
</tr>
<tr>
<td>PA-2T3/E3-EC</td>
<td>22-19</td>
</tr>
<tr>
<td>PA-8CE1</td>
<td>22-18</td>
</tr>
<tr>
<td>PA-A3-T3</td>
<td>22-19</td>
</tr>
<tr>
<td>PA-MC-4T1</td>
<td>22-18</td>
</tr>
<tr>
<td>SFP-OC3-IR1</td>
<td>22-17</td>
</tr>
<tr>
<td>SFP-OC3-SR</td>
<td>22-17</td>
</tr>
<tr>
<td>SFP-OC48-IR1</td>
<td>22-16</td>
</tr>
<tr>
<td>SPA-1XCHOC12/DS0</td>
<td>22-18</td>
</tr>
<tr>
<td>SPA-1XCHSTM1/OC3</td>
<td>22-18</td>
</tr>
<tr>
<td>SPA-1XOC12-ATM</td>
<td>22-19</td>
</tr>
<tr>
<td>SPA-1XOC12-ATM-V2</td>
<td>22-19</td>
</tr>
<tr>
<td>SPA-2XCT3/DS0</td>
<td>22-19</td>
</tr>
<tr>
<td>SPA-8X1FE-TX-V2</td>
<td>22-14, 22-15</td>
</tr>
<tr>
<td>SPA-8XCHT1/E1</td>
<td>22-18</td>
</tr>
<tr>
<td>SPA-8XOC12-POS</td>
<td>22-17</td>
</tr>
<tr>
<td>SPA-CHOC3-CE-ATM</td>
<td>22-18</td>
</tr>
<tr>
<td>SPA-OC192POS-LR</td>
<td>22-17</td>
</tr>
<tr>
<td>VIC-4VP-FXS-DID</td>
<td>22-22</td>
</tr>
<tr>
<td>VIC-EM</td>
<td>22-22</td>
</tr>
<tr>
<td>VWIC-1MFT-T1E1</td>
<td>22-18</td>
</tr>
<tr>
<td>VWIC-2MFT-T1-DIR</td>
<td>22-18</td>
</tr>
<tr>
<td>WIC-1ADSL</td>
<td>22-19</td>
</tr>
<tr>
<td>WIC-1ADSL-DG</td>
<td>22-19</td>
</tr>
<tr>
<td>WIC-1AM-V2</td>
<td>22-19</td>
</tr>
<tr>
<td>WIC-1B-U-V2</td>
<td>22-22</td>
</tr>
<tr>
<td>WIC-1SHDSL</td>
<td>22-19</td>
</tr>
<tr>
<td>WIC-SERIAL-1T</td>
<td>22-21</td>
</tr>
<tr>
<td>WS-6700-DFC3B</td>
<td>22-14, 22-15</td>
</tr>
<tr>
<td>WS-6700-DFC3BXL</td>
<td>22-14, 22-15</td>
</tr>
<tr>
<td>ws-c2924-xl-v</td>
<td>22-23</td>
</tr>
<tr>
<td>WS-C3560G-24TS</td>
<td>22-23</td>
</tr>
<tr>
<td>WS-F6K-MSFC2A</td>
<td>22-23</td>
</tr>
<tr>
<td>WS-SUP32-10GE-3B</td>
<td>22-14, 22-15</td>
</tr>
<tr>
<td>WS-SUP720-3BXL</td>
<td>22-23</td>
</tr>
<tr>
<td>WS-X4148-RJ</td>
<td>22-23</td>
</tr>
<tr>
<td>WS-X4624-SFP-E</td>
<td>22-14, 22-15</td>
</tr>
<tr>
<td>wsx6ksup1a2ge</td>
<td>22-23</td>
</tr>
<tr>
<td>wsx6ksup22ge</td>
<td>22-23</td>
</tr>
<tr>
<td>module template name</td>
<td></td>
</tr>
<tr>
<td>10Gigaethernet-Gigaethernet-FiberGE-fiberoptic -ethernet-default</td>
<td>22-15</td>
</tr>
<tr>
<td>A10GigaEthernet</td>
<td>22-15</td>
</tr>
<tr>
<td>DWDMA10GigaEthernet</td>
<td>22-15</td>
</tr>
<tr>
<td>EthernetChannelSwitchDefault</td>
<td>22-15</td>
</tr>
<tr>
<td>ethernetDefault</td>
<td>22-15</td>
</tr>
<tr>
<td>ethernet-default-over-optic</td>
<td>22-15</td>
</tr>
<tr>
<td>ethernetDefault-RJ45-or-FiberE1orGigabitTechnology2</td>
<td>22-15</td>
</tr>
<tr>
<td>ethernet-or-oc48-pos-default</td>
<td>22-15</td>
</tr>
<tr>
<td>ethernet-or-oc-pos-default3</td>
<td>22-15</td>
</tr>
<tr>
<td>GE-over-OC12-pos-default</td>
<td>22-15</td>
</tr>
<tr>
<td>GE-over-OC3-pos-default3</td>
<td>22-15</td>
</tr>
<tr>
<td>gigaEthernet</td>
<td>22-15</td>
</tr>
<tr>
<td>module templates</td>
<td></td>
</tr>
<tr>
<td>evaluating</td>
<td>19-9</td>
</tr>
</tbody>
</table>
N

Network Element (NE) 2-3

P

parsing operators/rules 5-1
  header and footer 5-1
  match 5-6
  parse integer 5-9
  remove lines 5-3
  replace 5-5
  select lines 5-4
  set 5-7
  substring 5-8

pluggable modules
  commands for managing 19-18
  deleting 19-8
  editing 19-7
  viewing 19-7

Prime Network
  overview 1-1

Prime Network Administration
  starting 10-1

S

Soft Properties 1-2
  about 1-2
  basic example 4-1
  creating 3-7
  debugging 3-17
  defining a soft property table 4-9
  defining a TCA for 4-8
  defining general parameters 3-7
  defining parsing parameters 3-9
  defining TCA parameters 3-14
  deleting 3-22
  editing 3-17

importing and exporting 3-22
Manager 2-1
  element properties table 3-4
  File menu 3-5
  properties panel 3-4
  Toolbar 3-5
  Tools menu 3-5
  window 3-3
  workflow 3-6
  opening the Manager 3-2
  parsing operators/rules 5-1
  publishing 3-19
  regular expressions 7-1
  table example 4-9
  TCA example 4-8
  testing parsing rules 3-14
  viewing 3-19

software versions
  enabling support 18-16
  viewing 18-16

standard modules
  editing 19-4
  enabling support 19-1
  viewing 19-3

syslogs
  adding support for 20-7

T

templates
  genericUVNE 22-1
  U-VNE 22-1

testing U-VNEs 18-36

test resources
  VCB 17-5

Threshold Crossing Alarm (TCA)
  defining 3-14
  Soft Property example 4-8

thresholds
about 2-2
triggers 6-1
traps
   enabling support for 20-2

U
U-VNE
   creating 18-1
   GUI representation 18-2
   Managing with CLI 18-23
   moving to production 18-44
   templates 22-1
   testing and certifying 18-36
   using CLI to create and manage 18-19

V
VCE
   deleting registry customizations 21-3
   deploying extensions 21-1
   filtering tables 17-3
   importing registry customizations 21-3
   log file 17-4
   Overview 17-1
   Quick Tour 17-2
   template reference 22-1
   test resources 17-5
   troubleshooting device type customization 18-43
VCE CLI
   Adding and Managing Device Types 18-31
   adding and managing U-VNEs 18-23
   commands for managing modules 19-10
   commands for managing pluggable modules 19-18
   customizing events 20-13
   introductory command reference 18-20
   managing modules 19-9
Virtual Network Element (VNE)
description 2-3
VNE Customization Builder
   customization
      exporting 21-3
   script
      VcbImportCommands.sh 21-3
   testing
      event customization 20-12
      resources for 17-5
      VCB log file, using 17-4
VNE drivers
   viewing and managing 18-11

W
workflow
   aborting 9-23, 10-3
   deleting 9-23, 10-4
   deleting a template 9-22, 10-2
   executing based on template 9-22
   getting output 9-23
   managing 10-1
   order of working with 8-4
   testing 9-20
   viewing output 10-2
   viewing templates 10-2
   viewing the history 10-4
Workflow Editor
   about 1-2, 8-1
   attribute types 9-5
   BQL tasks 8-2
   changing attribute types 9-5
   lock task 8-3
   synchronous workflow nesting 8-3
   toolbar 9-3
   window 9-3
   workflow tasks 8-2
   working with 9-1
   workflow properties
execute BQL task 9-9
lock task 9-16
subflow task 9-14
unlock task 9-18
viewing 9-6
workflow call task 9-11
working with the Task library 9-8

workflow template
creating 9-4
deleting 9-6
deploying 9-5
names 9-4
retrieving 9-5