Juniper Networks Network and Security Manager

API Guide

Release
2012.2

Published: 2014-11-20
Revision 3

Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, California 94089
USA
408-745-2000
www.juniper.net

Network and Security Manager API Guide

Revision History
January 2013—Revision 1
December 2013—Revision 2
November 2014—Revision 3

The information in this document is current as of the date on the title page.

END USER LICENSE AGREEMENT

The Juniper Networks product that is the subject of this technical documentation consists of (or is intended for use with) Juniper Networks software. Use of such software is subject to the terms and conditions of the End User License Agreement ("EULA") posted at http://www.juniper.net/support/eula.html. By downloading, installing or using such software, you agree to the terms and conditions of that EULA.

Table of Contents

About This Guide xi
Objectives xi
Audience xi
Conventions xi
Documentation xiii
Requesting Technical Support xiii
Self-Help Online Tools and Resources xiv
Opening a Case with JTAC xiv

Part 1 NSM API

Chapter 1 Overview 3
NSM API Features 3
NSM API Authentication and Authorization 4
NSM API Error Handling 4

Chapter 2 NSM API Operations 7
System Service API 7
Data Centric Service API 8
Data Centric Service XML Subtree Filter 9
Data Centric Service Operations 9
Job Service API 11
Log Service API 13

Part 2 API Data Types

Chapter 3 Data Objects 17
API Data Objects 17

Chapter 4 Common Message Data Types 21
SimpleRequestType and SimpleResponseType Data Types 21

Chapter 5 Security Data Model 23
NSM Policy 24
Security Rulebases 25
Backdoor (rb_backdoor_collection) 25
Exempt (rb_exempt_collection) 29
Firewall (rb_firewall_collection) 31
IDP (rb_idp_collection) 37
Multicast (rb_multicast_collection) 40
SYN Protector (rb_syndef_collection) 42
Traffic Anomalies (rb_tsig_collection) 45

Network Honeypot (rb_portfaker_collection) . . . 47
Service (service_collection) . . . 50
Address (address_collection_type) . . . 51
Schedule Object (scheduleobj_collection_type) . . . 52
Attack (attack_collection) . . . 53
Antivirus (avobj_collection) . . . 57
GTP (gtpobj_collection_type) . . . 58
DI Profile (DIProfile_collection_type) . . . 60
Global DIP (globaldip_collection) . . . 61
Global MIP (globalmpi_collection) . . . 62
Global VIP (globalvip_collection) . . . 63
URL Filter Object (urlfilter_collection) . . . 64
NAT-DIP (dip_collection) . . . 66
NAT-MIP (Mip_collection) . . . 67
NAT-VIP (vipSet) . . . 68
Static Route . . . 69
Gateway . . . 71
VPN (vpn_collection) . . . 72
USER (user_collection) . . . 73

Part 3 Using the NSM API from a Perl Client

Chapter 6 **Installing the Perl Client Environment . . . 77**

Installing the Perl Client Environment on Linux-Unix Machines . . . 77
Installing the Perl Client Environment on Windows Machines . . . 78
NSM Perl API Directory Structure . . . 79
Using a Perl Script to Access the NSM API . . . 79

Chapter 7 **Using the Perl Client to Access the NSM API . . . 83**

Login and Logout . . . 83

Chapter 8 **Using the API to Manage Shared Objects . . . 85**

Using the Perl Client Library with Address Objects . . . 85
Add Address Objects . . . 85
Replace an Address Object . . . 87
Rename Address Objects . . . 87
Read Address Objects . . . 88
Delete Address Objects . . . 89
Delete All Address Objects . . . 89
Using the Perl Client Library with Service Objects . . . 89
Add Service Objects . . . 90
Add Group-Global Service Objects . . . 91
Read Group-Global Service Objects . . . 92
Replace Group-Global Service Objects . . . 93
Delete All Group-Global Service Objects . . . 93
Using the Perl Client Library with Device Objects . . . 93
Read Device Objects . . . 94
Troubleshooting Common Perl API issues . . . 94

Part 4 Using the NSM API from a Java Client

Chapter 9 Using APIs for Authentication 99

Login 99
Logout 100

Chapter 10 Using APIs for Policy Management 101

Create a New Policy 101
Update an Existing Policy 103
Delete a Policy 104
Get a List of Policies 105
Get a Policy 106
Assign a Policy to a Device 106
Remove a Policy Assignment 107
Delete Rule in Policy 108

Chapter 11 Using APIs for Shared Object Management 111

Insert a Shared Object 111
Replace a Shared Object 112
Delete a Shared Object 114
Get a List of Shared Objects 114
Get a Shared Object 115
Associate Shared Object to its Group Object 116

Chapter 12 Using APIs for Job Management 117

Get a Job Result 117
Import a List of Devices 118
Update a List of Devices 119
Get a Configuration Summary 119
Get a Running Configuration 120
Get the Delta Configuration 121
Cancel a Job Request 121
Validate Device Configuration 122

Chapter 13 Using APIs for Device Management 125

Retrieve Domains 125
Retrieve the Device List in One Domain 125

Chapter 14 Using APIs for DeviceObj Management 127

Retrieve Device Config with Default Values 127
Modify Device Configuration 128
Manage VPN configuration in DeviceObj Container 130
Insert Interface Configuration 130
Replace Interface Configuration 131
Delete Interface Configuration 133
Insert Gateway Configuration 133
Replace Gateway Configuration 135
Delete Gateway Configuration 136
Insert VPN Configuration 137
Replace VPN Configuration 138
Delete VPN Configuration 139

Insert Route Configuration . . . 140
Replace Route Configuration . . . 141
Delete Route Configuration . . . 142
Manage VPN Configuration in a junos-es Container for Junos OS Devices . . . 143
Insert Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration . . . 143
Delete Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration . . . 144
Replace Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration . . . 145
Manage NAT Configuration in DeviceObj Container . . . 147
Insert NAT (MIP/DIP/VIP) Configuration . . . 147
Replace NAT (MIP/DIP/VIP) configuration . . . 149
Delete NAT (MIP/DIP/VIP) Configuration . . . 151
Admin User Management . . . 152
Insert Admin Configuration . . . 152
Replace Admin Configuration . . . 153
Delete Admin Configuration . . . 154
Modification of Management interface for Junos OS Devices . . . 155
Insert Management Interface Configuration . . . 155
Replace Management Interface Configuration . . . 156
Delete Management Interface Configuration . . . 157
Assign SNMP Name and SNMP Contact for Junos OS Devices . . . 158

Part 5 NSM API WSDLs

Chapter 15 **Job Service API WSDL . . . 163**
WSDL File . . . 163

Chapter 16 **System Service API WSDL . . . 173**
WSDL File . . . 173

Chapter 17 **Data Centric API WSDL . . . 179**
WSDL File . . . 179

Chapter 18 **Log Service API WSDL . . . 191**
WSDL File . . . 191

Part 6 Index

Index . . . 195

List of Figures

Part 1 NSM API

Chapter 1 Overview . . . 3

Figure 1: ErrorType Data Type . . . 5

Part 2 API Data Types

Chapter 3 Data Objects . . . 17

Figure 2: NSM API Data Objects . . . 18

Chapter 4 Common Message Data Types . . . 21

Figure 3: SimpleRequestType Data Type . . . 21
Figure 4: SimpleResponseType Data Type . . . 21

Chapter 5 Security Data Model . . . 23

Figure 5: NSM Policy . . . 24
Figure 6: Backdoor Rulebase . . . 26
Figure 7: Exempt Rulebase . . . 30
Figure 8: Firewall Rulebase . . . 32
Figure 9: Firewall policy_type . . . 32
Figure 10: IDP Rulebase . . . 38
Figure 11: Multicast Rulebase . . . 41
Figure 12: SYN Protector Rulebase . . . 43
Figure 13: Traffic Anomalies Rulebase . . . 45
Figure 14: Network Honeypot Rulebase . . . 48
Figure 15: Service Collection . . . 50
Figure 16: Address Collection . . . 52
Figure 17: Schedule Object . . . 53
Figure 18: Antivirus Collection . . . 57
Figure 19: GTP Collection . . . 59
Figure 20: DI Profile . . . 61
Figure 21: Global DIP Collection . . . 62
Figure 22: Global MIP Collection . . . 63
Figure 23: Global VIP Collection . . . 64
Figure 24: URL Filter Object Collection . . . 65
Figure 25: NAT- DIP Object Collection . . . 66
Figure 26: NAT- MIP Object Collection . . . 68
Figure 27: NAT- VIP Object Collection . . . 69
Figure 28: Static Route Object Collection . . . 70
Figure 29: Gateway Object Collection . . . 71
Figure 30: VPN Object Collection . . . 73
Figure 31: USER Object Collection . . . 74

Part 3 Using the NSM API from a Perl Client

Chapter 6 Installing the Perl Client Environment 77

Figure 32: NSM Perl API Directory Structure 79

List of Tables

About This Guide xi

Table 1: Text Conventions xii
Table 2: Syntax Conventions xii
Table 3: Network and Security Manager Publications xiii

Part 1 NSM API

Chapter 1 Overview 3

Table 4: ErrorType Data Types 5

Chapter 2 NSM API Operations 7

Table 5: System Service API Operations 7
Table 6: Data Centric API Operations 9
Table 7: Job Service API Operations 11
Table 8: Log Service API Operations 14

Part 2 API Data Types

Chapter 3 Data Objects 17

Table 9: API Data Objects 18

Chapter 4 Common Message Data Types 21

Table 10: SimpleRequestType and SimpleResponseType Definitions 22

Chapter 5 Security Data Model 23

Table 11: NSM Policy Data Elements 24
Table 12: Backdoor Rulebase Data Elements 27
Table 13: Exempt Rulebase Data Elements 30
Table 14: Firewall Data Elements 33
Table 15: IDP Rulebase Data Elements 38
Table 16: Multicast Rulebase Data Elements 41
Table 17: SYN Protector Rulebase Data Elements 43
Table 18: Traffic Anamolies Rulebase Date Elements 45
Table 19: Network Honeypot Rulebase Data Elements 48
Table 20: Service Collection Data Elements 51
Table 21: Address Collection Data Elements 52
Table 22: Schedule Object Data Elements 53
Table 23: Attack Collection Data Elements 56
Table 24: Antivirus Collection Data Elements 58
Table 25: GTP Collection Data Elements 59
Table 26: DIP Data Elements 61
Table 27: Global DIP Data Elements 62

Table 28: Global MIP Data Elements 63
Table 29: Global VIP Data Elements 64
Table 30: URL Filter Data Collection 65
Table 31: NAT DIP Data Elements 66
Table 32: NAT-MIP Data Elements 68
Table 33: NAT-VIP Data Elements 69
Table 34: Static Route Data Elements 70
Table 35: Gateway Data Elements 71
Table 36: VPN Data Elements 73
Table 37: USER Data Elements 74

About This Guide

This preface provides the following guidelines for using the *NSM API Guide* and related Juniper Networks, Inc. technical documents:

- Objectives on page xi
- Audience on page xi
- Conventions on page xi
- Documentation on page xiii
- Requesting Technical Support on page xiii

Objectives

This guide explains how to use the Network and Security Manager (NSM) API to manage device configurations and control communications between the API, external web clients, and the internal NSM GUI client.

Audience

This guide is written for developers and network administrators who configure and monitor Juniper Networks DMI and non-DMI compliant device routing platforms.

- Customers with technical knowledge of networks and the Internet.
- Network administrators who install, configure, and manage Juniper Networks products. Familiarity with the XML language is needed.

Conventions

The sample screens used throughout this guide are representations of the screens that appear when you install and configure the NSM software. The actual screens may differ.

All examples show default file paths. If you do not accept the installation defaults, your paths will vary from the examples.

Table 1 on page xii defines text conventions used in this guide.

Table 1: Text Conventions

Convention	Description	Examples
Bold typeface like this	• Represents commands and keywords in text. • Represents keywords • Represents UI elements	• Issue the **clock source** command. • Specify the keyword **exp-msg**. • Click **User Objects**
Bold typeface like this	Represents text that the user must type.	**user input**

`fixed-width font`	Represents information as displayed on the terminal screen.	`host1#`
		show ip ospf
		`Routing Process OSPF 2 with Router ID 5.5.0.250`
		`Router is an area Border Router (ABR)`

Key names linked with a plus (+) sign	Indicates that you must press two or more keys simultaneously.	Ctrl + d
Italics	• Emphasizes words • Identifies variables	• The product supports two levels of access, *user* and *privileged.* • *clusterID*, *ipAddress.*
The angle bracket (>)	Indicates navigation paths through the UI by clicking menu options and links.	**Object Manager > User Objects > Local Objects**

Table 2 on page xii defines syntax conventions used in this guide.

Table 2: Syntax Conventions

Convention	Description	Examples
Words in plain text	Represent keywords	terminal length
Words in italics	Represent variables	*mask, accessListName*
Words separated by the pipe (|) symbol	Represent a choice to select one keyword or variable to the left or right of this symbol. The keyword or variable can be optional or required.	diagnostic | line
Words enclosed in brackets ([ ])	Represent optional keywords or variables.	[internal | external]
Words enclosed in brackets followed by and asterisk ([ ]*)	Represent optional keywords or variables that can be entered more than once.	[level1 | level2 | 11]*
Words enclosed in braces ({ })	Represent required keywords or variables.	{ permit | deny } { in | out } { clusterId | ipAddress }

Documentation

Table 3 on page xiii describes documentation for the NSM.

Table 3: Network and Security Manager Publications

Book	Description
Network and Security Manager Installation Guide	Describes the steps to install the NSM management system on a single server or on separate servers. It also includes information on how to install and run the NSM user interface. This guide is intended for IT administrators responsible for the installation or upgrade of NSM.
Network and Security Manager Administration Guide	Describes how to use and configure key management features in the NSM. It provides conceptual information, suggested workflows, and examples. This guide is best used in conjunction with the NSM Online Help, which provides step-by-step instructions for performing management tasks in the NSM UI. This guide is intended for application administrators or those individuals responsible for owning the server and security infrastructure and configuring the product for multi-user systems. It is also intended for device configuration administrators, firewall and VPN administrators, and network security operation center administrators.
Network and Security Manager Configuring Screen OS and IDP Devices Guide	Describes NSM features related to device configuration and management. It also explains how to configure basic and advanced NSM functionality, including deploying new device configurations, managing security policies and VPNs, and general device administration.
Network and Security Manager Online Help	Provides procedures for basic tasks in the NSM user interface. It also includes a brief overview of the NSM system and a description of the GUI elements.
Network and Security Manager API Guide	Provides complete syntax and description of the SOAP messaging interface to NSM.
Network and Security Manager Release Notes	Provides the latest information about features, changes, known problems, resolved problems, and system maximum values. If the information in the Release Notes differs from the information found in the documentation set, follow the Release Notes. Release notes are included on the corresponding software CD and are available on the Juniper Networks Website.

Requesting Technical Support

Technical product support is available through the Juniper Networks Technical Assistance Center (JTAC). If you are a customer with an active J-Care or JNASC support contract, or are covered under warranty, and need postsales technical support, you can access our tools and resources online or open a case with JTAC.

- JTAC policies—For a complete understanding of our JTAC procedures and policies, review the *JTAC User Guide* located at http://www.juniper.net/us/en/local/pdf/resource-guides/7100059-en.pdf.
- Product warranties—For product warranty information, visit http://www.juniper.net/support/warranty/.
- JTAC Hours of Operation —The JTAC centers have resources available 24 hours a day, 7 days a week, 365 days a year.

Self-Help Online Tools and Resources

For quick and easy problem resolution, Juniper Networks has designed an online self-service portal called the Customer Support Center (CSC) that provides you with the following features:

- Find CSC offerings: http://www.juniper.net/customers/support/
- Find product documentation: http://www.juniper.net/techpubs/
- Find solutions and answer questions using our Knowledge Base: http://kb.juniper.net/
- Download the latest versions of software and review release notes: http://www.juniper.net/customers/csc/software/
- Search technical bulletins for relevant hardware and software notifications: http://kb.juniper.net/InfoCenter/
- Join and participate in the Juniper Networks Community Forum: http://www.juniper.net/company/communities/
- Open a case online in the CSC Case Management tool: http://www.juniper.net/cm/

To verify service entitlement by product serial number, use our Serial Number Entitlement (SNE) Tool: https://tools.juniper.net/SerialNumberEntitlementSearch/

Opening a Case with JTAC

You can open a case with JTAC on the Web or by telephone.

- Use the Case Management tool in the CSC at http://www.juniper.net/cm/.
- Call 1-888-314-JTAC (1-888-314-5822 toll-free in the USA, Canada, and Mexico).

For international or direct-dial options in countries without toll-free numbers, visit us at http://www.juniper.net/support/requesting-support.html

PART 1

NSM API

This part introduces the Network and Security Manager (NSM) Application Programming Interface (API) with a brief overview, summary of the required client environment, list of the component APIs, and examples.

- Overview on page 3
- NSM API Operations on page 7

CHAPTER 1

Overview

This section provides general information about the Network and Security Manager (NSM) API.

- NSM API Features on page 3
- NSM API Authentication and Authorization on page 4
- NSM API Error Handling on page 4

NSM API Features

The NSM API provides programmatic access to NSM and enables third-party developers to create applications that leverage the power of NSM. The API supports Simple Object Access Protocol/Hypertext Transmission Protocol Secure (SOAP/HTTPS). The SOAP API is built on open standards such as SOAP and the Web Service Definition Language (WSDL) supported by a range of development tools. You can use a third-party SOAP development tool to generate programming language objects and stubs from the WSDL that specifies the message schema. Your application works with data in the format of generated objects; it sends and receives the data by invoking the methods of stubs.

The API provides a rich set of data models for devices and security policies. The models are published in the format of XML schema (XSD).

In this release, the NSM API provides the following features and functions:

- Central policy management
- NSM object management
- NSM directives:
  - Import devices
  - Update device
  - Summarize delta configuration
  - Get running configuration
- Retrieve device list per domain
- Retrieve high level device status
- Retrieve log packet data

The CommonDataTypes.xsd file contains definitions of the data types described in this chapter.

For more information, see the *NSM Release Notes*, *NSM Administration Guide*, and *NSM Online Help* for client and server setup requirements.

This chapter contains the following sections:

NSM API Authentication and Authorization

Before the API can connect to the NSM server, a user must log into the NSM server using a user name, password, and domain name. This is analogous to the user sign in a regular GUI client. The application includes the authentication token in the subsequent API call requests to the NSM server.

NSM API Error Handling

If the API client encounters an error,either the client receives an error message or an exception is thrown. Two types of errors are possible.

- Application-level errors result from problems with application-level data on the client side or on the server side.
  - The request is missing a required field. In this case, the request is not sent out from the client side.
  - The request is valid, but a problem occurred when NSM processed the data.
- Infrastructure errors can occur on the client side or server side. The NSM application-level software does not catch this type of error, so exceptions are thrown by the API client code. The possible errors are:
  - NSM server is down
  - Problem with the client-side or server-side SOAP framework
  - Wrong server address

The NSM server catches all application-level errors and returns the error messages.

The result of a service request is either Success or Failure If a request fails, an error code and error message are returned as part of the response message.

Figure 1 on page 5 shows the basic structure of application-level errors returned by the NSM server. Table 4 on page 5 describes the frequently used ErrorType data type.

Figure 1: ErrorType Data Type

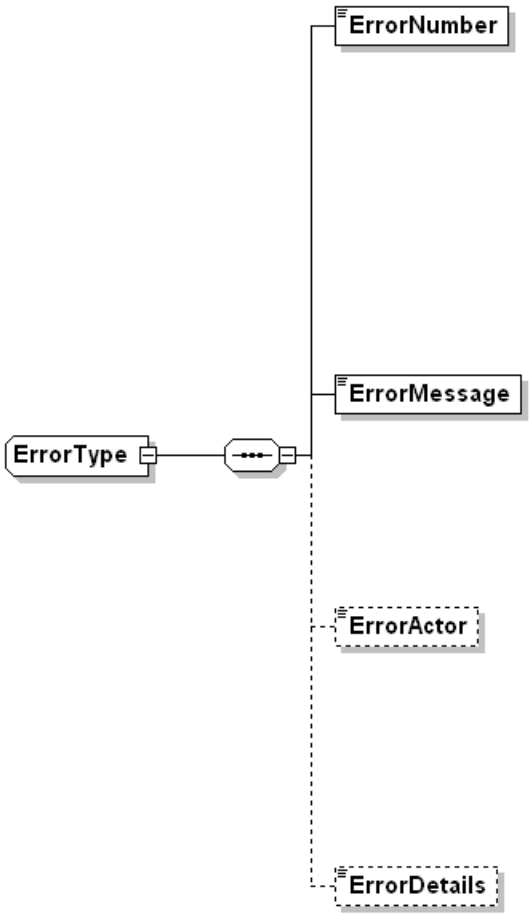


Table 4: ErrorType Data Types

Data Type	Description
ErrorType	These request errors (not infrastructure errors) are issued when the system encounters business data problems (for example, an invalid combination of arguments). This complexType data has the following sequence: • ErrorNumber = Unique number that identifies the particular error condition (type = unsignedInt). This data element is only used by the server. • ErrorMessage = Brief description of the condition that raised the error (type = string). • ErrorActor = The source (location) of the error (type = string). • ErrorDetails = Detailed error message (type = string).

CHAPTER 2

NSM API Operations

The application programming interface (API) defined by NSM is used to provision policies, manage and monitor devices, and generate reports. The API has four parts:

- System Service API on page 7
- Data Centric Service API on page 8
- Job Service API on page 11
- Log Service API on page 13

System Service API

The API System Service processes log in, log out, and system information requests.Table 5 on page 7 summarizes the API data elements.

For information about the WSDL file defining the API, see "System Service API WSDL" on page 173 .

Table 5: System Service API Operations

Operation	Description
LoginRequest	Log into NSM server. Request: • domainName = Domain supplied during login. The user logs in to this domain. NOTE: Use **global.<subdomain name>** to log in to a subdomain and **global** to log in to a global domain. • userName = User name supplied during login. • password = Password supplied during login. Response: • loginStatus = Uses LoginStatusCodeType to return: "Success" if the login is successful. "Failure" for login rejection. "Challenge" if the login request is being challenged but not yet denied. • authToken =Token returned for login request success. This token is reused for other requests during the current session.

Table 5: System Service API Operations *(continued)*

RespondToChallengeRequest	Reuses the token received in LoginResponse to send a response to the challenge. Receives a token if the response is successful. Request: Answer to the challenge question. Response: Token received.
LogoutRequest	Logout from the system. Request: none Response: none
GetSystemInfoRequest	This operation retrieves system information (service list and all accessible domain IDs and names). Request: serviceName Response: serviceDesc, domain name and domain ID.

NOTE: When using the LoginRequest API, enter global.<subdomain name> to log in to a particular subdomain. The login fails if just the subdomain name is used.

Data Centric Service API

The Data Centric Service API provides access to the internal data of NSM. It receives incoming data access requests, retrieves the data from NSM, conducts any necessary transformations, and sends the transformed data back as responses. This section introduces the XML subtree filter used with the service and describes the Data Centric data elements.

The following features are supported in the current release of NSM.

- Write access to DeviceObj
- Accessing default values of screenOs devices
- The support for identifying Object Collisions will be provided in the NSM server code. Only duplicate object names will be verified during Insert Operation
- The support for logging the Success or Failure Test cases in Audit Log manager

NOTE: In the current release of NSM, write access to the deviceobj and sysvpn from the Data Centric Service is blocked to protect data integrity.

See "Data Centric API WSDL" on page 179 for a description of the API-defining WSDL file.

Data Centric Service XML Subtree Filter

The filter used in the Data Centric Service API is the XML subtree filter defined by NETCONF. Subtree filtering is a mechanism that allows an application to select particular XML subtrees from the configurations from the devices.

A subtree filter consists of zero or more element subtrees, which represent the filter selection criteria.

Five types of components may be present in a subtree filter:

- Namespace selection
- Attribute matching expressions
- Containment nodes
- Selection nodes
- Content matching nodes

Only the first four are supported in the NSM API.

Data Centric Service Operations

Table 6 on page 9 summarizes Data Centric Service operations.

Table 6: Data Centric API Operations

Operations	Description
GetObjectDependentRequest	Gets objects that refer to the object specified in the request. Request: • objectIdentifier= Identifies the object to be retrieved (type = objectIdentifierType) • dbVersionId= Version of the database (type = unsignedInt) • objectFilter = Filter to be applied to the result (type = ObjectFilterType) • metadataOnly = If true, only the metadata is returned. Otherwise, the entire object is returned. Response: object
GetObjectViewByCategoryRequest	Gets objects in one category. Request: • category = Schema name of the category (type = string) • domainId = Domain of the schema (type = unsignedShort) • dbVersionId= Version of the database (type = unsignedInt) • objectFilter = Filter to be applied to the result (type = ObjectFilterType) • view = Transformation of the object. For the default view, the returned object follows the schema with no transformation (type = ViewType). • property = Transformation parameters (type = NameValueType). Response: object

Table 6: Data Centric API Operations *(continued)*

GetObjectViewByIdRequest	Gets objects by ID. Request: • objectIdentifier= Identifies the object to be retrieved (type = objectIdentifierType) • dbVersionId= Version of the database (type = unsignedInt) • objectFilter = Filter to be applied to the result (type = ObjectFilterType) • view = Transformation of the object. For the default view, the returned object follows the schema with no transformation (type = ViewType). • property = Transformation parameters (type = NameValueType). Response: object
LockObjectRequest	Locks the specified object. Request: objectIdentifier Response: objectLockStatus
UnlockObjectRequest	Unlocks the specified object. Request: object Response: objectLockStatus
ModifyObjectViewRequest	Modifies the object. All commands in the request are executed in one transaction. ModifyObjectViewRequest supports the following operations: • Update Node • Insert node before / after • Append node • Insert Object • Replace Object • Delete Object NOTE: You should lock the object before modifying it and unlock it afterwards. The modification will fail if the object is locked by a different user session. However, a modification request without prior locking can run if the object is not locked by the others. Data corruption does not occur even if an API user forgets to lock an object before modifying it. Request: • command = Command that modifies the object (type = ModifyCommand). Response: • metadata • objectModification • subObjectModification

Table 6: Data Centric API Operations *(continued)*

QueryObjectViewRequest	Queries the object. Request: • category = Schema name of the category (type = string). • simpleQuery = Query expression (type = SimpleQueryType). • dbVersionId= Version of the database (type = unsignedInt) • objectFilter = Filter to be applied to the result (type = ObjectFilterType) • view = Transformation of the object. For the default view, the returned object follows the schema with no transformation (type = ViewType). • property = Transformation parameters (type = NameValueType). Response: • queryId • object
ResolveObjectReferenceRequest	Resolves the object reference. Request: • objectReference = Object reference to be resolved (type = string) • dbVersionId= Version of the database (type = unsignedInt) • objectFilter = Filter to be applied to the result (type = ObjectFilterType) Response: object

Job Service API

The Job Service API processes command directives to configure devices and display the results. Table 7 on page 11 summarizes the API operations.

See "Job Service API WSDL" on page 163 for the Web Service Description Language (WSDL) definition of the API.

Table 7: Job Service API Operations

Operation	Description
UpdateDeviceRequest	Request to update the device configuration. Request: JobRequestType • jobName= Name of the job. • scheduleTime= Time when the job will run. If not specified, the job will run immediately. • jobArgs= List of the devices to which the job applies (type = JobArgsType). Response: JobResponseType • status = Job status. • jobName = Name of the job. • response = Response to the job.

Table 7: Job Service API Operations *(continued)*

Operation	Description
ImportDeviceRequest	Request to import a device configuration from physical devices. Request: JobRequestType • jobName= Name of the job. • scheduleTime= Time when the job will run. If not specified, the job will run immediately. • jobArgs= List of the devices to which the job applies. Response: JobResponseType • status = Job status. • jobName = Name of the job. • response = Response to the job.
GetConfigSummaryRequest	Request for a summary of the configuration currently running on a physical device. Request: JobRequestType • jobName= Name of the job. • scheduleTime= Time when the job will run. If not specified, the job will run immediately. • jobArgs= List of the devices to which the job applies. Response: JobResponseType • status = Job status. • jobName = Name of the job. • response = Response to the job.
GetRunningConfigRequest	Request for the configuration currently running on a physical device. Request: JobRequestType • jobName= Name of the job. • scheduleTime= Time when the job will run. If not specified, the job will run immediately. • jobArgs= List of the devices to which the job applies. Response: JobResponseType • status = Job status. • jobName = Name of the job. • response = Response to the job.

Table 7: Job Service API Operations *(continued)*

Operation	Description
GetDeltaConfigRequest	Request for the differences between the modeled device configuration and the configuration currently running on a physical device. Request: JobRequestType • jobName= Name of the job. • scheduleTime= Time when the job will run. If not specified, the job will run immediately. • jobArgs= List of the devices to which the job applies. Response: JobResponseType • status = Job status. • jobName = Name of the job. • response = Response to the job.
GetJobStatusRequest	Request for the status of a job. Request: • domainName = Name of the domain associated with the job. • domainId= ID of the domain. • jobName= Name of the job. Response: JobResponseType • status = Job status. • jobName = Name of the job. • response = Response to the job.
GetJobResultRequest	Request for status of a completed job. Request: • domainName = Name of the domain associated with the job. • domainId= ID of the domain. • jobName= Name of the job. Response: JobResponseType • status = Job status. • jobName = Name of the job. • response = Response to the job.

Log Service API

The Log Service API retrieves and displays logs of NSM events. Table 8 on page 14 summarizes these operations.

See "Data Centric API WSDL" on page 179 for the WSDL file defining the API.

Table 8: Log Service API Operations

Operation	Description
GetPacketDataRequest	Gets both the log data and the packet data that triggers the log. Request: • dayId= Identifier for the day. • recordNum= Record number. Response: • numPackets= Number of packets returned. • triggerPacket = Packet triggering the log event. • data = Log data.

PART 2

API Data Types

- Data Objects on page 17
- Common Message Data Types on page 21
- Security Data Model on page 23

CHAPTER 3

Data Objects

- API Data Objects on page 17

API Data Objects

A data object is data that is identifiable by the NSM API. NSM data objects are logically grouped by domains and categories. A domain is a logical grouping of devices, their security policies, and their access privileges. A category is a logical grouping of the data objects that have the same structure. For example, a deviceobj is a category of data objects for devices.

Each object in a category is identified by a tuple (domain, category, object id). The XML representation of the data objects conforms to the XML schemas illustrated later in this chapter.

Objects are referenced by name or ID.

- Reference based on id. The reference to an object is defined in the following format

  &<domain id>.<category name>.<object id>. For example, **&1.service.100**.

- Reference based on name. The reference to an object defined in the following format **&<domain id>.<category name>.?????????<object name>**. Here, nine question marks precede **<object name>**.

The key data types are illustrated in Figure 2 on page 18. The entire set of common data types is described in Table 9 on page 18.

Figure 2: NSM API Data Objects

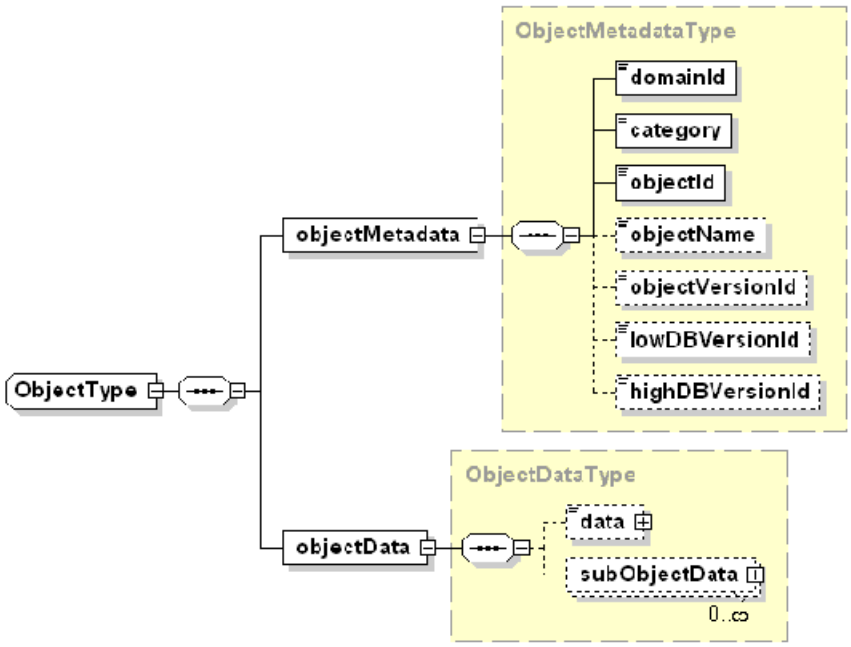


Table 9: API Data Objects

Data Type	Description
DataFormatType	This simpleType data code has the following possible enumeration values: • "XML" • "NML" • "XML_FROM_NML" • "NML_AND_XML" • "JAVA_OBJECT" • "FILE"
OpaqueDataType	This complexType data code (base64Binary element) has one value: • attribute name = "dataFormat" (type = DataFormatType)
ObjectIdentifierType	This complexType data code has the following sequence: • domainId = ID of the domain (type = unsignedShort) • category = Schema name of the category (type = string) • objectIdOrName = Object ID or name (type = ObjectIdOrNameType).

Table 9: API Data Objects *(continued)*

Data Type	Description
ObjectMetadataType	This complexType data code has the following sequence: • domainId = ID of the domain (type = unsignedShort) • category = Schema name of the category (type = string) • objectId = Object identifier (type = unsignedInt). • objectName = Name of the object (type = string). • objectVersionId = Identifier of the object version (type = unsignedInt). • lowDBVersionId= Lowest database version identifier (type = unsignedInt). • highDBVersionId= Highest database version identifier (type = unsignedInt).
ObjectIdOrNameType	This complexType data code takes only one of the following inputs: • objectId = Object identifier (type = unsignedInt). or • objectName = Name of the object (type = string).
DomainIdOrNameType	This complexType data code takes only one of the following inputs: • objectId = Object identifier (type = unsignedInt). or • objectName = Name of the object (type = string).
SubObjectDataType	This complexType data code has the following sequence: • subCategory = Subcategory under “category” (type = string). • data = Subobject data (type = OpaqueDataType).
ObjectDataType	This complexType data code has the following sequence: • objectName = Name of the object (type = string). • data = Object data (type = OpaqueDataType). • subObjectData = Data of the subobject (type = SubObjectDataType).
ObjectType	This complexType data code has the following sequence: • objectMetadata = Metadata object (type = ObjectMetadataType). • objectData = Object data (type = ObjectDataType).
StatusCodeType	This simpleType data code has the following possible enumeration values: • Success = Request is successful. • Failure = Request has failed.
AuthTokenType	This complexType data code is the security header for SOAP API calls. It has one value: • Token = String identifying the user (type = string).
SequenceType	This complexType data code is a partial response type. It has the following sequence: • SequenceNum= Sequence number of the current response (type = int). • IsDone = Total number of response messages (type = Boolean).

Table 9: API Data Objects *(continued)*

Data Type	Description
ProgressType	This complexType data code takes one input: • CompletionPercent = Percent completed of the response (type = unsignedInt).
ConversationContextType	This complexType data code has the following sequence: • ConversationId = Identifier for the message conversation (type = string). • UserSessionContext = Describes the context of the user session (type = anyType). • AuditLogContext = Context for the audit log (type = anyType). • ACFilter = Filter (type = anyType).

CHAPTER 4

Common Message Data Types

This chapter describes the message types, SimpleRequest and SimpleResponse, that are most commonly used in API data messages.

- SimpleRequestType and SimpleResponseType Data Types on page 21

SimpleRequestType and SimpleResponseType Data Types

The frequently used data types SimpleRequestType and SimpleResponseType are illustrated in Figure 3 on page 21 and Figure 4 on page 21. They are described in Table 10 on page 22

Figure 3: SimpleRequestType Data Type

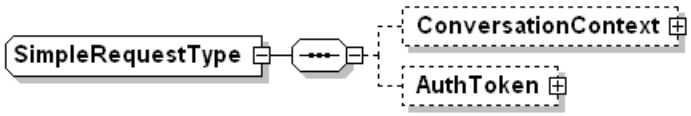


Figure 4: SimpleResponseType Data Type

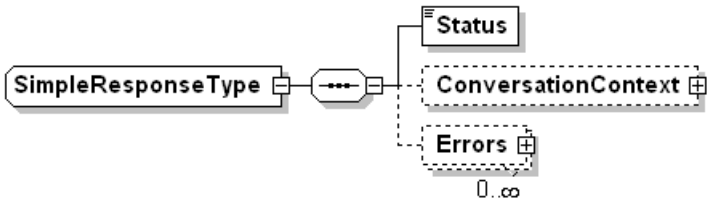


Table 10: SimpleRequestType and SimpleResponseType Definitions

Data Type	Description
SimpleRequestType	Base type definition of the SOAP body of the request. All request types are derived from the abstract type. The naming convention for concrete type names is the name of the service (verb or call name) followed by "RequestType." Generally, VerbNameRequestType. This complexType data has the following sequence: • ConversationContext = Context of the message conversation (type = ConversationContextType). • AuthToken = Token returned for the simple request (type = AuthTokenType).
SimpleResponseType	Base type definition of the SOAP body of a response. This complexType data has the following sequence: • Status = Status of the response (type = StatusCodeType). • ConversationContext = Context of the message conversation (type = ConversationContextType). • Errors = Errors returned (type = ErrorType).

CHAPTER 5

Security Data Model

This chapter introduces aspects of the API data model that apply to NSM security policies.

For complete details, see the **dm.xsd** and **dm.xsd** definition files included with the file set in this release.

The **adm.xsd** is located at **$NSROOT/GuiSvr/var/be/schemas/dmi-nsm/** . The smaller zip file (**adm.zip**) is located at **$NSROOT/GuiSvr/var/be/schemas/dmi-nsm/document**.

This chapter contains the following sections:

- NSM Policy on page 24
- Security Rulebases on page 25
- Service (service_collection) on page 50
- Address (address_collection_type) on page 51
- Schedule Object (scheduleobj_collection_type) on page 52
- Attack (attack_collection) on page 53
- Antivirus (avobj_collection) on page 57
- GTP (gtpobj_collection_type) on page 58
- DI Profile (DIProfile_collection_type) on page 60
- Global DIP (globaldip_collection) on page 61
- Global MIP (globalmpi_collection) on page 62
- Global VIP (globalvip_collection) on page 63
- URL Filter Object (urlfilter_collection) on page 64
- NAT-DIP (dip_collection) on page 66
- NAT-MIP (Mip_collection) on page 67
- NAT-VIP (vipSet) on page 68
- Static Route on page 69
- Gateway on page 71
- VPN (vpn_collection) on page 72
- USER (user_collection) on page 73

NSM Policy

The NSM Policy collection (**nsmpolicy_collection**) data elements are illustrated and described in Figure 5 on page 24 and Table 11 on page 24.

Figure 5: NSM Policy

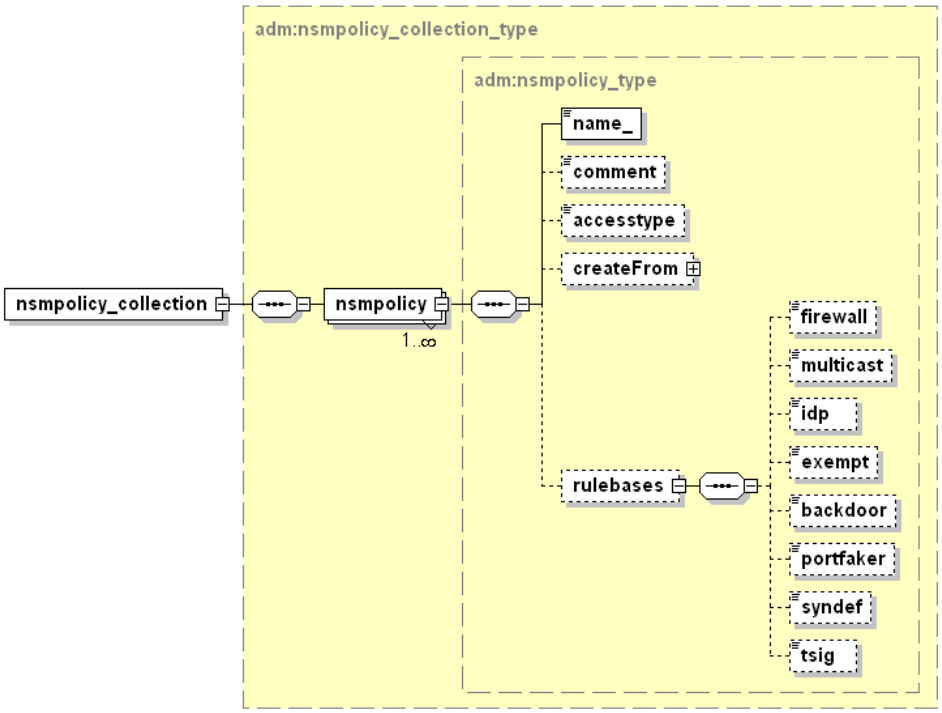


Table 11: NSM Policy Data Elements

Data Element	Description
name_	Name of the security policy (string).
comment	Comments about the security policy.
accesstype	Type of access. (enum) Possible values are: • regular = regular policy • pre = domain pre policy • post = domain post policy • mompre = central manager pre policy • mompost = central manager post policy
createFrom	(Optional) Effective start date for the NSM security policy.
rulebases	Collection of references of rulebases. For more information, see "Security Rulebases" on page 25.

Table 11: NSM Policy Data Elements *(continued)*

Data Element	Description
firewall	Reference of the firewall rulebase. Firewall rule data elements are included in a security policy. For more information, see "Firewall (rb_firewall_collection)" on page 31.
multicast	Reference of the multicast rulebase. Multicast rule data elements are included in a security policy. For more information, see "Multicast (rb_multicast_collection)" on page 40.
idp	Reference of the IDP rulebase, Idp rule data elements are included in a security policy. For more information, see "IDP (rb_idp_collection)" on page 37.
exempt	Reference of the Exempt rulebase. Exempt rule data elements are included in a security policy. For more information, see "Exempt (rb_exempt_collection)" on page 29.
backdoor	Reference of the backdoor rulebase. Backdoor rule data elements are included in a security policy. For more information, see "Backdoor (rb_backdoor_collection)" on page 25.
portfaker	Network Honeypot (portfaker) rulebase. These data elements are included in a security policy. For more information, see "Traffic Anomalies (rb_tsig_collection)" on page 45.
syndef	Reference of the SYN Protector rulebase, These data elements are included in a security policy. For more information, see "SYN Protector (rb_syndef_collection)" on page 42.
tsig	Traffic Anomalies rulebase. These data elements are included in a security policy. For more information, see "Traffic Anomalies (rb_tsig_collection)" on page 45.

Security Rulebases

NSM security policies are configured by applying rules that are grouped into rulebases. Each rulebase can contain one or more rules, which are statements that define specific types of network traffic. When traffic passes through a security device, the device attempts to match that traffic against its list of rules. If a rule is matched, the device performs the action defined in the rule against the matching traffic. Zone rules enable traffic to flow between zones (interzone) or between two interfaces bound to the same zone (intrazone). Global rules are valid across all zones available on the device. Security devices process rules in the zone-specific rulebase first, and then rules in the global rulebase.

The NSM API data model supports the security policy rulebases summarized in the following sections.

Backdoor (rb_backdoor_collection)

The backdoor rulebase collection (rb_backdoor_collection) contains rules that enable NSM to detect attempted backdoor intrusions. A backdoor is a mechanism installed on a host computer that enables unauthorized access to the system. Attackers who have already compromised a system can install a backdoor to make future attacks easier. When attackers type commands to control a backdoor, they generate interactive traffic. Unlike antivirus software, which scans for known backdoor files or executables on the host system, IDP detects the interactive traffic that is produced when backdoors are

used. If interactive traffic is detected, IDP can perform IP actions against the connection to prevent the attacker from further compromising your network.

When you configure a backdoor rule, you must specify the following:

- Source and destination addresses for traffic that will be monitored
- Services that are offered by the source or destination and any interactive services that can be installed and used by attackers

For configuration procedures, see the *NSM Online Help* and the *NSM Administrator's Guide*.

The data elements in the backdoor rulebase are illustrated and described in Figure 6 on page 26 and Table 12 on page 27.

Figure 6: Backdoor Rulebase

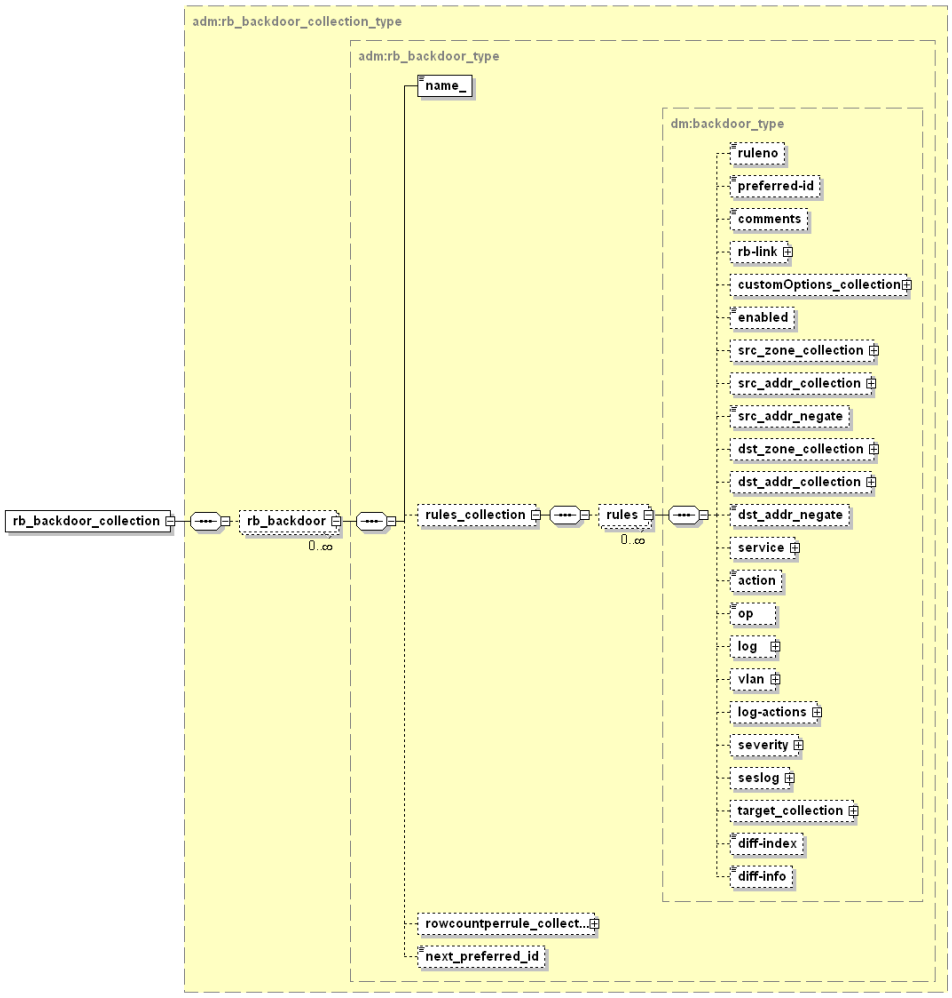


Table 12: Backdoor Rulebase Data Elements

Data Element	Description
name_	Name of the backdoor rule type. (string).
rules_collection	Collection of all sets of rules.
rules	Collection of all rules.
ruleno	Rule number.
preferred-id	A rule ID is a number that uniquely identifies a rule within the rulebase and security policy. After you install a rule as part of a security policy on a security device, you can view the rule by logging in locally to the device. However, when you view it through the Web UI or CLI, the rule appears as an individual policy. The individual policy on the device has the same ID as the rule in the management system, enabling you to determine which rules are on specific devices.
comments	Comments about the backdoor rules.
rb-link	Rule group name.
customOptions_collection	Custom options.
enabled	Collection enabled.
src_zone_collection	The source sends traffic from this zone.
src_addr_collection	Address of the traffic source.
src_addr_negate	Negates the specified source address.
dst_zone_collection	The source sends traffic to this zone.
dst_addr_collection	Destination address for the traffic.
dst_addr_negate	Negates the specified destination address.
service	These service object rules specify the service that an attack uses to access the network.

Table 12: Backdoor Rulebase Data Elements *(continued)*

Data Element	Description
action	For each attack that matches a rule, you can choose an action that will occur if the IDP detects interactive traffic. The following actions are possible: • Accept = IDP accepts the interactive traffic • Drop Connection = IDP drops the interactive connection without sending an RST packet to the sender. This prevents the traffic from reaching its destination. This action is selected to drop connections from traffic that is not prone to spoofing. • Close Client = IDP closes the interactive connection to the client but not to the server. • Close Server = IDP closes the interactive connection to the server but not to the client. • Close Client and Server = IDP closes the interactive connection and sends a RST packet to both the client and the server. If IDP is operating in an inline tap mode, IDP sends a RST packet to both the client and the server but does not close the connection.
op	Sets the operation to detect or ignore. If you select detect, choose an action to perform if backdoor traffic is detected.
log	If this parameter is enabled, the API logs an attack and creates log records with attack information. You can display this information real time in the Log Viewer. For more critical attacks, you can set an alert flag that will appear in the log record.
vlan	This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are: • Any (default) = Any rule will be applied to messages in any VLAN and to messages without a VLAN tag. This setting has the same effect as not specifying a VLAN. Any can be sent to devices that do not support VLAN tagging. • None = A rule will be applied only to messages that do not have a VLAN tag. Rules with this value set cannot be sent to devices that do not support VLAN tagging. • vlan_list_collection = Specifies the VLAN tags to which the rule applies. You must create VLAN objects before applying them to the rules. Rules with this value set cannot be sent to devices that do not support VLAN tagging.
log-actions	Action to be taken on the log. This can include configuring SNMP, Syslog, CSV, XML, script, and e-mail settings.

Table 12: Backdoor Rulebase Data Elements *(continued)*

severity	Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings: • Default • Info • Warning • Minor • Major • Critical
seslog	Log packets.
target_collection	Specifies the security devices or templates that will receive and use this rule. You can select multiple security devices on which to install the rule.

Exempt (rb_exempt_collection)

The exempt (rb_exempt_collection) rulebase works in conjunction with the IDP rulebase. Before you create exempt rules, you must create rules in the IDP rulebase. If traffic matches a rule in the IDP rulebase, IDP attempts to match the traffic against the rules in the exempt rulebase before performing the specified action or creating a log record for the event. When the IDP rulebase is deleted, the exempt rulebase is automatically deleted. When you create an exempt rule, you must specify the source and destination traffic to be exempted and the specific attacks that IDP will exempt.

The data elements in the exempt rulebase are illustrated and described in Figure 7 on page 30 and Table 13 on page 30.

Figure 7: Exempt Rulebase

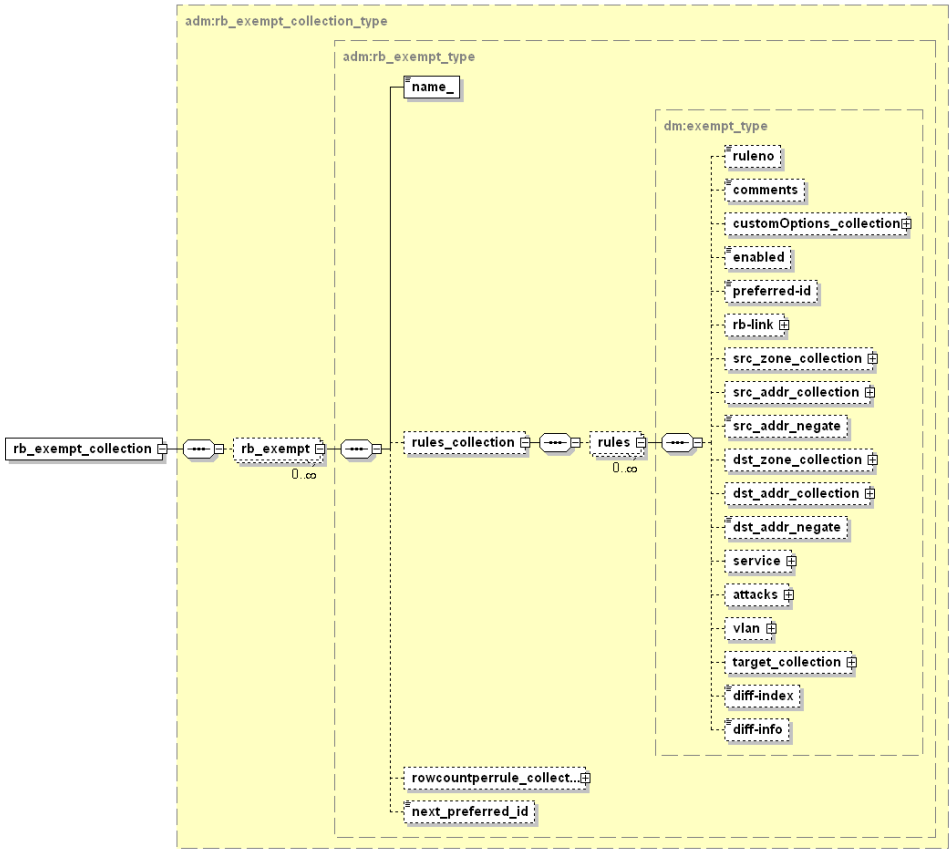


Table 13: Exempt Rulebase Data Elements

Data Element	Description
name_	Name of the exempt type.
rules_collection	Collection of all sets of rules.
rules	Collection of all rules.
rowcountperrule_collection	Row count per rule in the collection.
next_preferred_id	Next preferred ID.
ruleno	Rule number.
comments	Comments about the exempt collection.
customOptions_collection	Custom options.
enabled	Collection enabled.

Table 13: Exempt Rulebase Data Elements *(continued)*

Data Element	Description
preferred-id	A rule ID is a number that uniquely identifies a rule within the rulebase and security policy. After you install a rule as part of a security policy on a security device, you can view the rule by logging in locally to the device. However, when you view it through the Web UI or CLI, the rule appears as an individual policy. The individual policy on the device has the same ID as the rule in the management system, enabling you to determine which rules are on specific devices.
rb-link	Rule group name.
src_zone_collection	The source sends traffic from this zone.
src_addr_collection	Address of the traffic source.
src_addr_negate	Negates the specified source address.
dst_zone_collection	The source sends traffic to this zone.
dst_addr_collection	Destination address for the traffic.
dst_addr_negate	Negates the specified destination address.
service	Exempt type service.
attacks	The attacks that IDP will exempt for the specified source/destination address. You must include at least one attach object in an exempt rule.
vlan	This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are: • Any (default) = Any rule will be applied to messages in any VLAN and to messages without a VLAN tag. This setting has the same effect as not specifying a VLAN. Any can be sent to devices that do not support VLAN tagging. • None = A rule will be applied only to messages that do not have a VLAN tag. Rules with this value set cannot be sent to devices that do not support VLAN tagging. • vlan_list_collection = Specifies the VLAN tags to which the rule applies. You must create VLAN objects before applying them to the rules. Rules with this value set cannot be sent to devices that do not support VLAN tagging.
target_collection	Specifies the security devices or templates that will receive and use this rule. You can select multiple security devices on which to install the rule.

Firewall (rb_firewall_collection)

The firewall (rb_firewall_collection) rulebase contains zone-specific and global rules. A security policy can contain two firewall rulebases: zone-specific and global.

The data elements in the firewall rulebase are illustrated and described in Figure 8 on page 32, Figure 9 on page 32, and Table 14 on page 33.

Figure 8: Firewall Rulebase

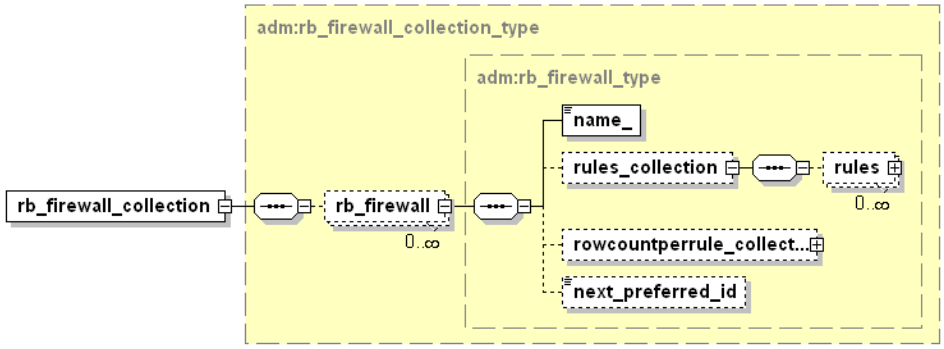


Figure 9: Firewall policy_type

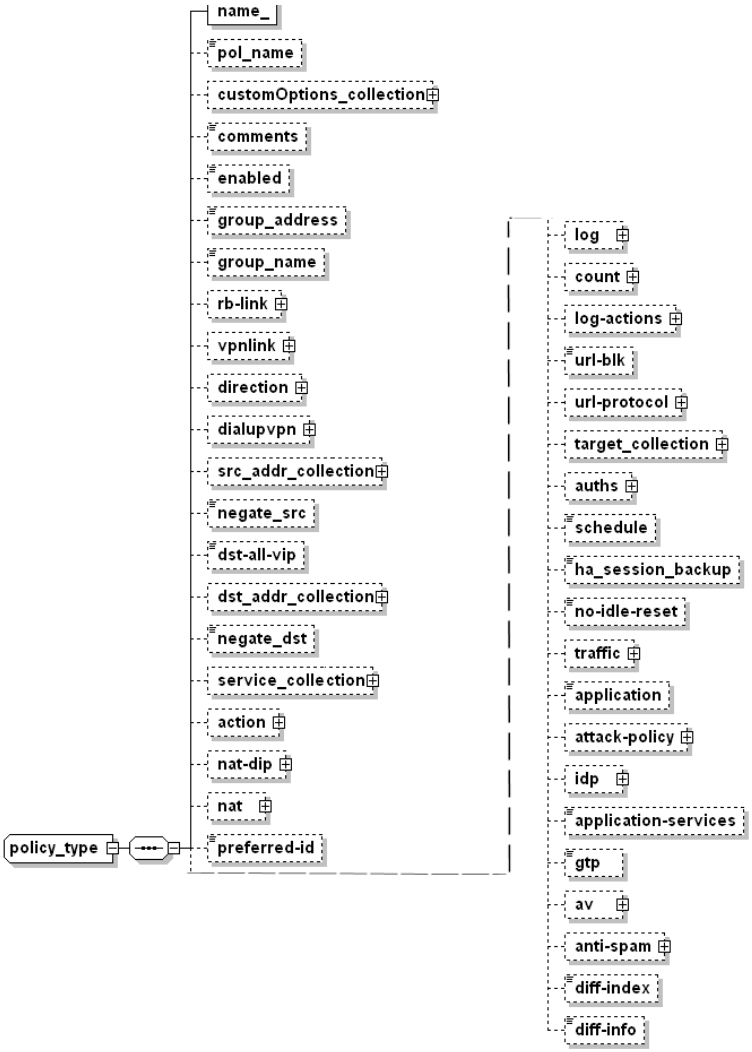


Table 14: Firewall Data Elements

Data Element	Description
name_	Name of the policy type.
rules_collection	Collection of all sets of rules.
rowcountperrule_collection	Row count per rule in the collection.
next_preferred_id	Next preferred ID.
pol_name	Rule title.
customOptions_collection	Custom options.
comments	Comments about the firewall collection.
enabled	Collection enabled.
group_address	Rule group ID. (string)
rb-link	Rule group name.
vpnlink	VPN link associated with the policy type.
direction	Policy direction.
dialupvpn	RAS VPN.
src_addr_collection	Address of the traffic source.
negate_src	Negates the specified source.
dst-all-vip	All VIPs. (Boolean)
dst_addr_collection	Destination address for the traffic.
negate_dst	Negates the specified destination.
service_collection	Configure the services supported by the destination. If the service of the network traffic matches a service selected in the rule, the security device performs the action that you select in the Action column. For more information, see "Service (service_collection)" on page 50.
action	Determines the action to be performed by the security device when it detects traffic that matches the rule. The possible values are: • deny • permit • reject • tunnel

Table 14: Firewall Data Elements *(continued)*

Data Element	Description
nat-dip	Source NAT.
nat	You can configure your security device to perform policy level network address translation (NAT) for any zone to translate the source address of incoming and outgoing traffic. You can configure the firewall to select a new source address from a Dynamic IP pool (DIP). For incoming traffic only, use a Mapped IP (MIP).
preferred-id	A preferred-id is a rule ID, a number that uniquely identifies a rule within the rulebase and security policy. After you install a rule as part of a security policy on a security device, you can view the rule by logging in locally to the device. However, when you view it through the Web UI or CLI, the rule appears as an individual policy. The individual policy on the device has the same ID as the rule in the management system, enabling you to determine which rules are on specific devices.
log	Deep inspection alert log.
count	Select Counting if you want to count how many bytes the matching network traffic contains and view this information in other applications. Possible values: • disabled • enabled
log-actions	Action to be taken on the log. This can include configuring SNMP, Syslog, CSV, XML, script, and e-mail settings.
url-blk	Web filtering. (default = false)
url-protocol	URL protocol.
target_collection	Specifies the security devices or templates that will receive and use this rule. You can select multiple security devices on which to install the rule.

Table 14: Firewall Data Elements *(continued)*

Data Element	Description
auths	You must include HTTP, FTP, or Telnet service objects in the Service column of the rule to enable remote users to authenticate themselves using Authentication. You can include other services as well, or specify all services. If authentication succeeds, the NSM allows the remote user to establish a connection to the destination address. If authentication fails, NSM drops the initial connection. If the source address supports multiple remote user accounts (for example, a Unix host running Telnet) or it is located behind a NAT device that uses a single IP address for all NAT assignments, only the first remote user from that source address must initiate and authenticate an HTTP, FTP, or Telnet connection. All subsequent remote users from that source address do not have to authenticate, and can pass matching network traffic to the destination address. If you use WebAuth, to make a connection to the destination address in the rule, the remote user must first initiate an HTTP connection to the WebAuth server. Your security device responds with a login prompt. After the remote user provides a user name and password, NSM attempts to authenticate the user credentials. If authentication succeeds, NSM permits the remote user to establish a connection to the destination address. If authentication fails, NSM drops the initial connection. The possible values: • no-auth • infranet-auth • auth • webauth
schedule	You can determine when a security device applies a rule to network traffic by defining a schedule for the rule.
ha_session_backup	If you select **HA Session Backup**, a rule with the Permit action will not be active when the session switches to the modern link. When this happens, the rule takes the Deny action.
no-idle-reset	Disable modem idle timer reset. (default = false)
traffic	Traffic shaping enables you to control the amount of bandwidth that is available to the matching network traffic in a rule. It also enables you to set a priority that determines how the security device handles matching network traffic that exceeds the defined maximum bandwidth. For security devices running ScreenOS 5.3 or later, you can also manage the flow of traffic through the security device by limiting bandwidth at the incoming point. The possible values: • gbw • priority • mbw

Table 14: Firewall Data Elements *(continued)*

Data Element	Description
application	Application. NOTE: You can override a service that is set in the Service column at the application layer. The service set in the Service column remains in force for the transport layer. Possible enumeration values: • DNS • FTP • HTTP • IMAP • SMTP • POP3 • H245 • Q931 • RAS • PORTMAPPER • SIP • SQLNETV2 • TALK • TFTP • REAL • RTSP • VDO • XING • IGNORE • MGCP_CA • MGCP_UA • PPTP • RSH • SCCP • AIM • YMSG • SMB • MSN • NBNAME • NBDS • NAS • NONE = No application specified. (default)
attack-policy	Security devices running ScreenOS 5.3 or later support Deep Inspection. A Deep Inspection (DI) Profile object contains predefined attack object groups (created by Juniper Networks) or your own custom attack object groups.
idp	Intrusion Detection and Prevention (IDP) is only supported on devices that have an IDP license installed. When you install a IDP license, DI is disabled on the device.

Table 14: Firewall Data Elements *(continued)*

Data Element	Description
application-services	Application services. Possible values: • None (default) • RWX • RRWX
gtp	You can use a GTP object in a firewall rule to determine how your security devices handles GTP traffic
av	To detect viruses in network traffic, you can configure the rule to forward traffic to an antivirus scanner. The server or Scan Manager returns the traffic—after it is cleaned or altered—and the security device executes the action specified in the Action column.
anti-spam	Antispam.

IDP (rb_idp_collection)

The IDP (rb_idp_collection) rulebase includes IDP rules that protect your network from attacks by using attack objects to identify malicious activity and take action. When you create an IDP rule, you specify the type of network traffic to be monitored for attacks including the from and to zone, source IP and destination IP for the network traffic, and service (type of IP traffic associated with the application layer protocols supported at the destination IP address). In security policies, service objects define the type of traffic that a rule must monitor.

These data elements are illustrated and described in Figure 10 on page 38 and Table 15 on page 38.

Figure 10: IDP Rulebase

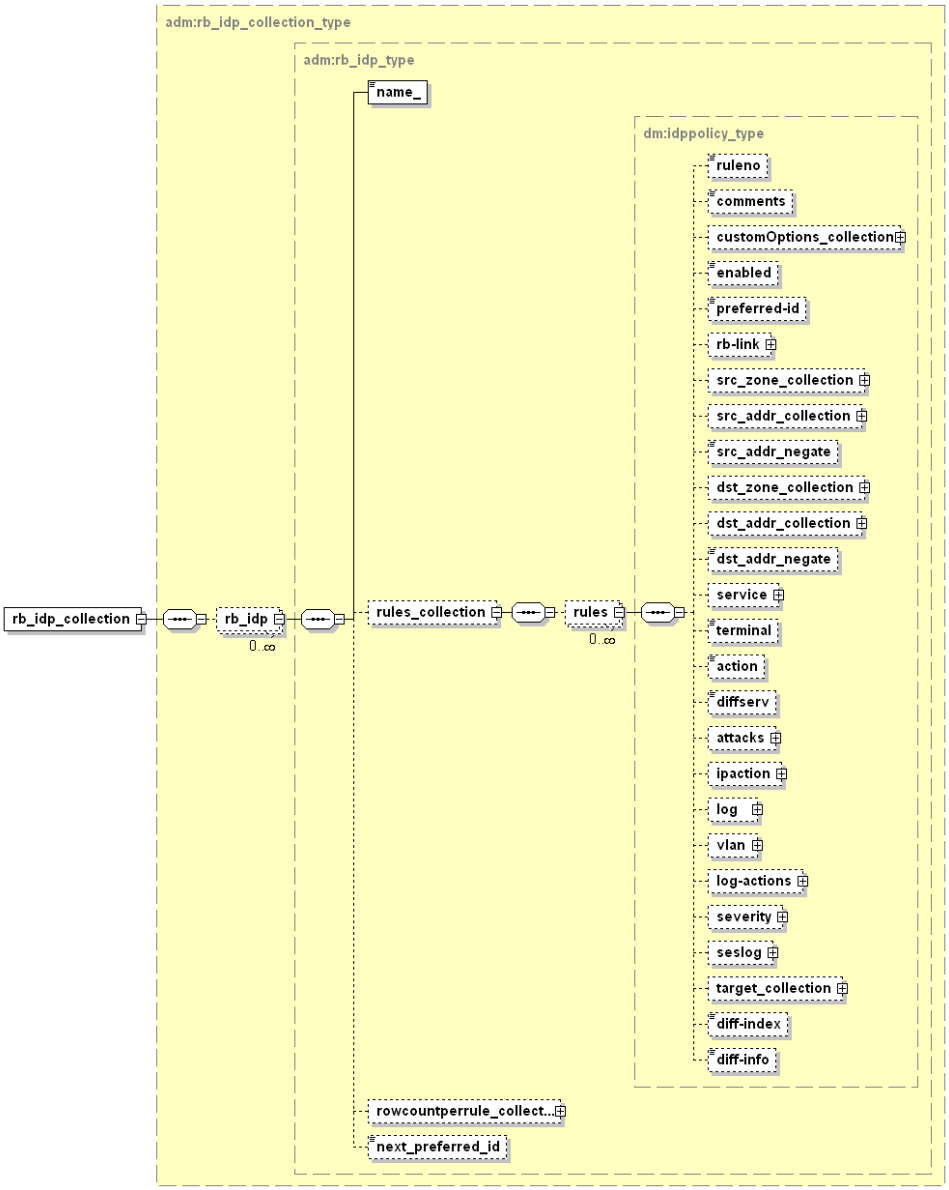


Table 15: IDP Rulebase Data Elements

Data Element	Description
name_	Name of the IDP collection.
rules_collection	Collection of all sets of rules.
rowcountperrule_collection	Row count per rule in the collection.
next_preferred_id	Next preferred ID.

Table 15: IDP Rulebase Data Elements *(continued)*

Data Element	Description
ruleno	Rule number.
comments	Comments about the IDP collection.
customOptions_collection	Custom options.
enabled	Collection enabled.
preferred-id	A rule ID is a number that uniquely identifies a rule within the rulebase and security policy. After you install a rule as part of a security policy on a security device, you can view the rule by logging in locally to the device. However, when you view it through the Web UI or CLI, the rule appears as an individual policy. The individual policy on the device has the same ID as the rule in the management system, enabling you to determine which rules are on specific devices.
rb-link	Rule group name.
src_zone_collection	The source sends traffic from this zone.
src_addr_collection	Address of the traffic source.
src_addr_negate	Negates the specified source address.
dst_zone_collection	The source sends traffic to this zone.
dst_addr_collection	Destination address for the traffic.
dst_addr_negate	Negates the specified destination address.
service	Application layer protocols that are supported by the destination IP address.
terminal	Makes a rule terminal. Traffic matching the source, destination, and service of a terminal rule is not compared to subsequent rules even if the traffic does not match an attack object in the terminal rule.
action	For each attack that matches a rule, you can choose an action that will occur if the IDP detects interactive traffic. The following actions are possible: • Accept = IDP accepts the interactive traffic • Drop Connection = IDP drops the interactive connection without sending a RST packet reset flag) to the sender. This prevents the traffic from reaching its destination. This action is selected to drop connections from traffic that is not prone to spoofing. • Close Client = IDP closes the interactive connection to the client but not to the server. • Close Server = IDP closes the interactive connection to the server but not to the client. • Close Client and Server = IDP closes the interactive connection and sends a RST packet to both the client and the server. If IDP is operating in an inline tap mode, IDP sends a RST packet to both the client and the server but does not close the connection.
diffserv	DiffServ Marking.

Table 15: IDP Rulebase Data Elements *(continued)*

Data Element	Description
attacks	Attack objects represent specific patterns of malicious activity within a connection. They also specify a method for detecting attacks.
ipaction	Enables and configures an IP action to prevent future malicious connections from the attacker's IP address.
log	Deep inspection alert log
vlan	This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are: • Any (default) = Any rule will be applied to messages in any VLAN and to messages without a VLAN tag. This setting has the same effect as not specifying a VLAN. Any can be sent to devices that do not support VLAN tagging. • None = A rule will be applied only to messages that do not have a VLAN tag. Rules with this value set cannot be sent to devices that do not support VLAN tagging. • vlan_list_collection = Specifies the VLAN tags to which the rule applies. You must create VLAN objects before applying them to the rules. Rules with this value set cannot be sent to devices that do not support VLAN tagging.
log-actions	Action to be taken on the log. This can include configuring SNMP, Syslog, CSV, XML, script, and e-mail settings.
severity	Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings: • Default • Info • Warning • Minor • Major • Critical
seslog	Log packets.
target_collection	Specifies the security devices or templates that will receive and use this rule. You can select multiple security devices on which to install the rule.

Multicast (rb_multicast_collection)

The multicast (rb_multicast_collection) rulebase includes multicast rules. Multicast rules are statements that define specific types of multicast control traffic. When multicast control traffic passes through a security device, the device attempts to match that traffic against its list of rules. If a rule is matched, the device performs the action defined in the rule against the matching traffic.

By default, security devices do not permit multicast control traffic (such as IGMP and PIM-SM messages) to cross security devices. However, you can secure device multicast control traffic through access lists. You can create an access list that defines the multicast

groups that hosts can join or to restrict the sources from which traffic is received, then reference these access lists in multicast rules. To enable multicast control traffic to pass between zones, you must configure multicast rules that specify the source zone (that sends out multicast traffic), multicast group sending out the traffic, destination zone for the traffic, and optionally, the destination group (source multicast group mapped to another multicast group address).

These data elements are illustrated and described in Figure 11 on page 41 and Table 16 on page 41.

Figure 11: Multicast Rulebase

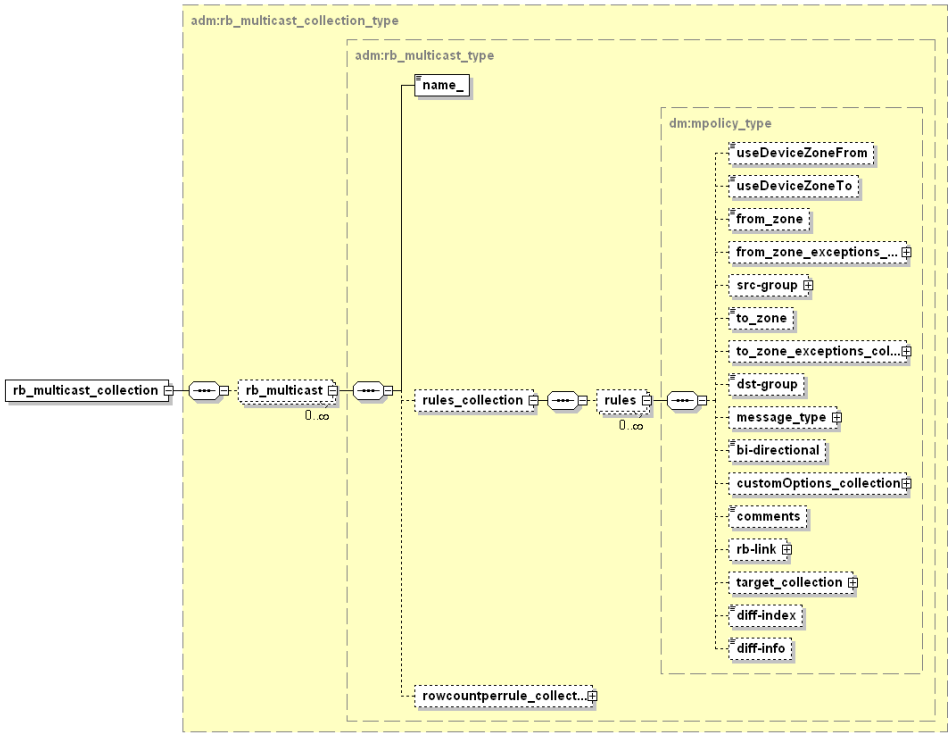


Table 16: Multicast Rulebase Data Elements

Data Element	Description
rb_multicast	Multicast rules.
name_	Rules (string). Name of the rulebase collection.
rules_collection	Collection of all sets of rules.
rowcountperrule_collection	Collection of row count per rules.
useDeviceZoneFrom	Marks the start point for the zone in which to use the device.
useDeviceZoneTo	Marks the end point for the zone in which to use the device.

Table 16: Multicast Rulebase Data Elements *(continued)*

Data Element	Description
from_zone	You must select a single zone for the source zone. The source will send multicast traffic from this zone.
from_zone_exceptions	From_zone exceptions.
src-group	Multicast group(s) to which the multicast traffic is sent. Multicast policy rules define the flow of multicast traffic between the source and multicast groups. You can use this parameter to specify one particular multicast group, any multicast group, or an access list that identifies the allowed multicast groups.
to_zone	Destination zone. The source will send multicast traffic to this zone.
to_zone_exceptions_collection	To_zone exceptions.
dst-group	Destination group. Optionally, you can map the source multicast group address to another multicast group address. When the source sends the multicast traffic to a multicast group address, the security device translates the original multicast group address to another address that you specified.
message_type	The rule applies to this type of multicast control traffic.
bi-directional	Bi-directional policy.
customOptions_collection	Custom options.
comments	Comment about the multicast collection.
rb-link	Rule group name.
target_collection	Specifies the security devices or templates that will receive and use this rule. You can select multiple security devices on which to install the rule.

SYN Protector (rb_syndef_collection)

The SYN Protector (rb_syndef_collection) rulebase protects your network from SYN-floods by ensuring that a three-way handshake is performed successfully for specified TCP traffic. If you know that your network is vulnerable to a SYN-flood, use the SYN-Protector rulebase to prevent it.

These data elements are illustrated and described in Figure 12 on page 43 and Table 17 on page 43.

Figure 12: SYN Protector Rulebase

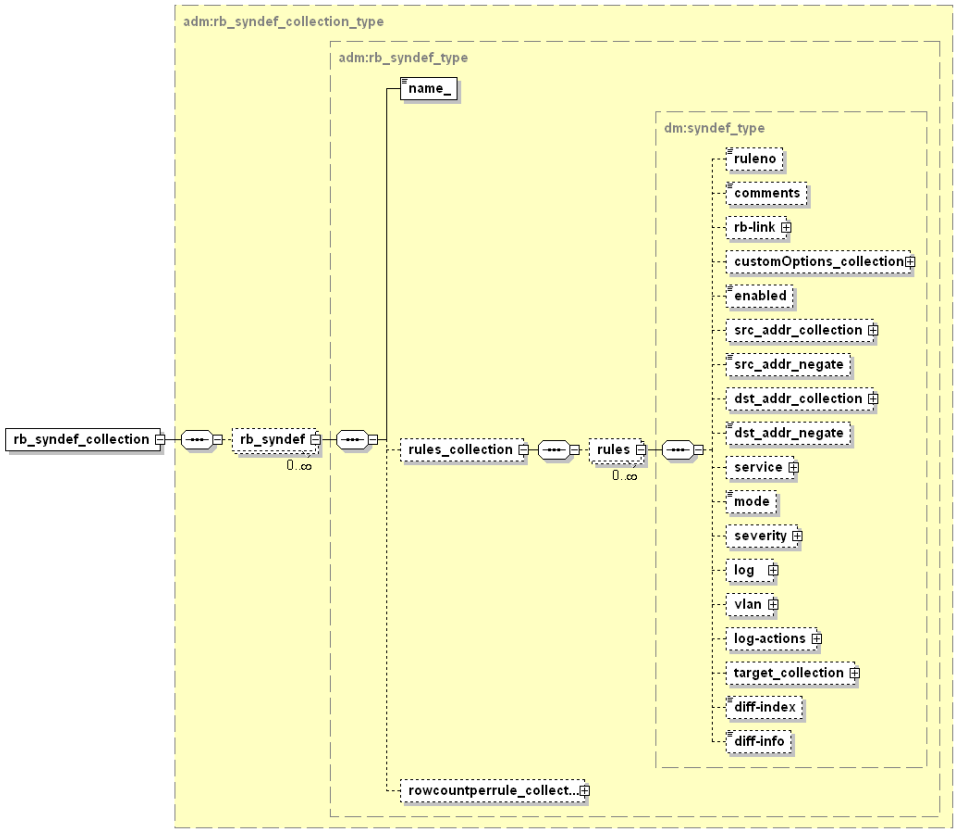


Table 17: SYN Protector Rulebase Data Elements

Data Element	Description
rb_syndef	SYN Protector rules.
name_	Name of SYN Protector rule.
rules_collection	Collection of all sets of rules.
rowcountperrule_collection	Row count per rule in the collection.
rules	Collection of all rules.
ruleno	Rule number.
comments	Comments about the SYN Protector collection.
customOptions_collection	Custom options.
enabled	Collection enabled.
src_addr_collection	Traffic source address.

Table 17: SYN Protector Rulebase Data Elements *(continued)*

Data Element	Description
src_addr_negate	Negates the specified source address.
dst_addr_coillection	Traffic destination address.
dst_addr_negate	Negates the specified destination address.
service	The default service, **TCP-any**, looks for SYN floods in all TCP-based traffic. NOTE: Always set the SYN Protector service value to **TCP-any**. Selecting individual services can cause unpredictable interactions with other rulebases.
mode	Select the mode that indicates how IDP handles TCP traffic. The possible values are: • None = no action taken. • Relay = IDP acts as the middleman or relay for the established connection. • Passive = IDP handles the transfer of packets between the client host and the server but does not prevent the connection from being established.
severity	Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings: • Default • Info • Warning • Minor • Major • Critical
log	You can configure the system to log an attack and create log records with attack information. This logged information can be viewed in real-time through the Log Viewer.
vlan	This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are: • Any (default) = Any rule will be applied to messages in any VLAN and to messages without a VLAN tag. This setting has the same effect as not specifying a VLAN. Any can be sent to devices that do not support VLAN tagging. • None = A rule will be applied only to messages that do not have a VLAN tag. Rules with this value set cannot be sent to devices that do not support VLAN tagging. • vlan_list_collection = Specifies the VLAN tags to which the rule applies. You must create VLAN objects before applying them to the rules. Rules with this value set cannot be sent to devices that do not support VLAN tagging.
log-actions	Action to be taken on the log. This can include configuring SNMP, Syslog, CSV, XML, script, and e-mail settings.
target_collection	Specifies the security devices or templates that will receive and use this rule. You can select multiple security devices on which to install the rule.

Traffic Anomalies (rb_tsig_collection)

The traffic anomalies (rb_tsig_collection) rulebase protect your network from attacks by using traffic flow analysis to identify attacks that occur over multiple connections and sessions (such as scans).

These data elements are illustrated and described in Figure 13 on page 45 and Table 18 on page 45.

Figure 13: Traffic Anomalies Rulebase

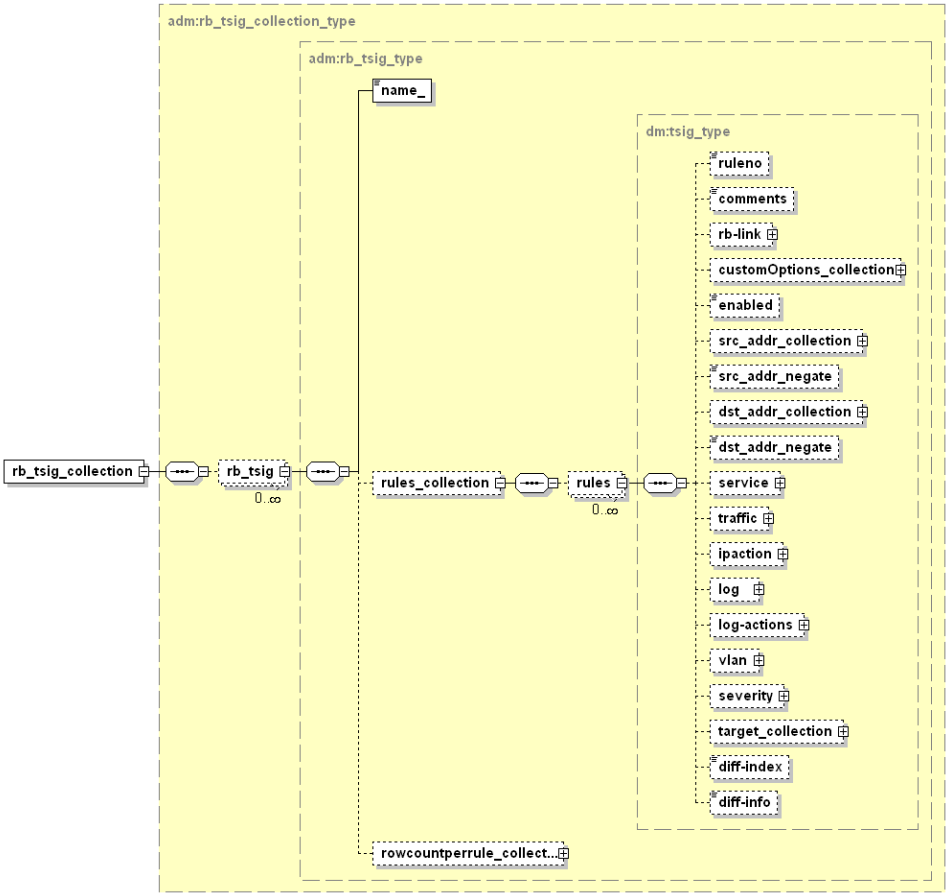


Table 18: Traffic Anamolies Rulebase Date Elements

Data Element	Description
rb_tsig	Traffic anomalies rules.
name_	Name of the traffic rule collection.
rules_collection	Collection of all sets of rules.
rowcountperrule_collection	Row count per rule in the collection.

Table 18: Traffic Anamolies Rulebase Date Elements *(continued)*

Data Element	Description
ruleno	Rule number.
comments	Comments about the traffic anomalies collection.
rb-link	Rule group name.
customOptions_collection	Custom options.
enabled	Collection enabled.
src_addr_collection	Address of the traffic source.
src_addr_negate	Negates the specified source address.
dst_addr_collection	Destination address for the traffic.
dst_addr_negate	Negates the specified destination address.
service	Service
traffic	Specifies how IDP will treat matching traffic. Possible values: • Ignore = The IDP Sensor ignores the traffic. • Detect = The IDP Sensor detects the traffic but does not log it. • TCP and UDP Port Scans = The IDP Sensor logs a TCP or UDP Port Scan, recording the TCP or UDP ports and a time interval during which the IDP Sensor records count of that number of TCP or UDP ports. For example, assume that the Port Count is 4 and the Time Threshold is 2 seconds. If the IDP Sensor monitors 4 TCP or UDP ports over 2 seconds from the same source IP to the same destination IP, the IDP Sensor logs it as a TCP or UDP port scan. • Distributed Port Scan = The IDP Sensor logs a Distributed Port Scan, recording the count of unique IP addresses and a time interval during which the IDP Sensor records count of that number of number of distributed addresses or ports. For example, the IP Count is 4 and the Time Threshold is 2 seconds. If the IDP Sensor monitors 4 IP addresses over 2 seconds from the same source IP to the same destination IP, the IDP Sensor logs it as a distributed port scan. • ICMP Sweep = The IDP Sensor logs an ICMP Sweep, recording the count of unique IP addresses and a time interval during which the IDP Sensor records count of that number of IP addresses. For example, the IP Count is 4 and the Time Threshold is 2 seconds. If the IDP Sensor monitors 4 IP addresses over 2 seconds from the same source IP, the IDP Sensor logs it as an ICMP sweep. • Network Scan = The IDP Sensor logs a Network Scan, recording the count of unique IP addresses and a time interval during which the IDP Sensor records count of that number of IP addresses. For example, the IP Count is 4 and the Time Threshold is 2 seconds. If the IDP Sensor monitors 4 IP addresses over 2 seconds from the same source IP, the IDP Sensor logs it as a network scan.
ipaction	Enables and configures an IP action to prevent future malicious connections from the attacker's IP address.
log	GTP logging.

Table 18: Traffic Anamolies Rulebase Date Elements *(continued)*

Data Element	Description
log-actions	Log action settings. Possible settings include configuring: • SNMP • Syslog • CVS • XML • script • e-mail
vlan	This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are: • Any (default) = Any rule will be applied to messages in any VLAN and to messages without a VLAN tag. This setting has the same effect as not specifying a VLAN. Any can be sent to devices that do not support VLAN tagging. • None = A rule will be applied only to messages that do not have a VLAN tag. Rules with this value set cannot be sent to devices that do not support VLAN tagging. • vlan_list_collection = Specifies the VLAN tags to which the rule applies. You must create VLAN objects before applying them to the rules. Rules with this value set cannot be sent to devices that do not support VLAN tagging.
severity	Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings: • Default • Info • Warning • Minor • Major • Critical
target_collection	Specifies the security devices or templates that will receive and use this rule. You can select multiple security devices on which to install the rule.

Network Honeypot (rb_portfaker_collection)

The network honeypot rulebase (rb_portfaker_collection) protects your network by impersonating open ports on existing servers on your network and alerting you to attackers performing port scans and other information-gathering activities.

These data elements are illustrated and described in Figure 14 on page 48 and Table 19 on page 48.

Figure 14: Network Honeypot Rulebase

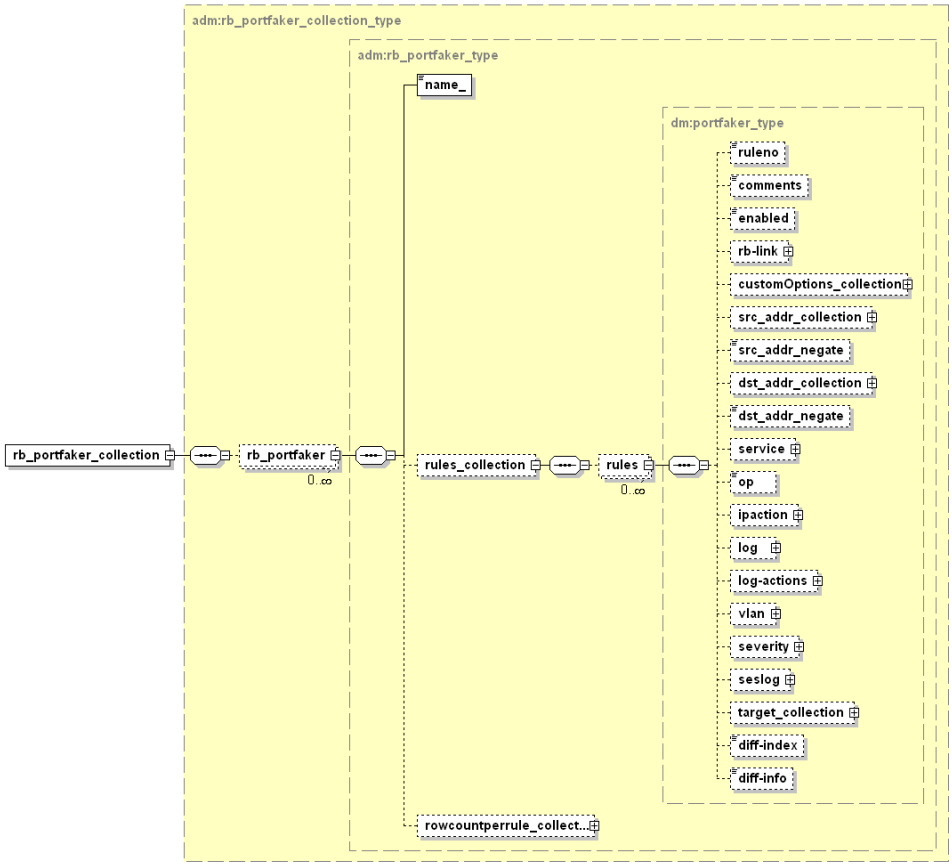


Table 19: Network Honeypot Rulebase Data Elements

Data Element	Description
rb_portfaker	Network honeypot (portfaker) rules.
name_	Name of the portfaker type.
rules_collection	Collection of all sets of rules.
rowcountperrule_collection	Row count per rule in the collection.
rules	Collection of all rules.
ruleno	Rule number
comments	Comments about the network honeypot (portfaker) collection.
enabled	Collection enabled.
rb_link	Portfaker Link collection

Table 19: Network Honeypot Rulebase Data Elements *(continued)*

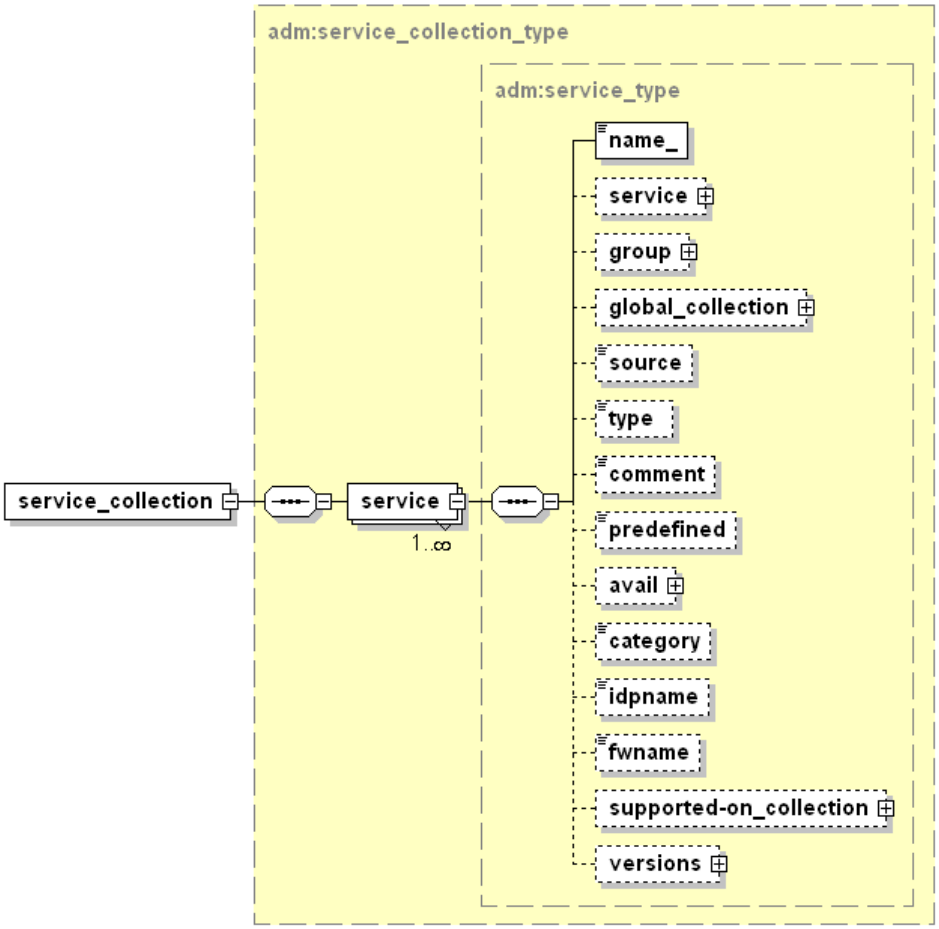
Data Element	Description
customOptions_collection	Custom options.
src_addr_collection	Address of the traffic source.
src_addr_negate	Negates the specified source address.
dst_addr_collection	Destination address for the traffic.
dst_addr_negate	Negates the specified destination address.
service	Service
op	Operation
ipaction	Enables and configures an IP action to prevent future malicious connections from the attacker's IP address.
log	Logging
log-actions	Action to be taken on the log. This can include configuring SNMP, Syslog, CSV, XML, script, and e-mail settings.
vlan	This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are: • Any (default) = Any rule will be applied to messages in any VLAN and to messages without a VLAN tag. This setting has the same effect as not specifying a VLAN. Any can be sent to devices that do not support VLAN tagging. • None = A rule will be applied only to messages that do not have a VLAN tag. Rules with this value set cannot be sent to devices that do not support VLAN tagging. • vlan_list_collection = Specifies the VLAN tags to which the rule applies. You must create VLAN objects before applying them to the rules. Rules with this value set cannot be sent to devices that do not support VLAN tagging.
severity	Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings: • Default • Info • Warning • Minor • Major • Critical
seslog	Log packets.
target_collection	Specifies the security devices or templates that will receive and use this rule. You can select multiple security devices on which to install the rule.

Service (service_collection)

The service collection (service_collection) defines services. These services represent the types of IP traffic that are associated with protocol standards. In a security policy, a service object defines the type of traffic that the rule will monitor. Related services are aggregated into service groups.

These data elements are illustrated and described in Figure 15 on page 50 and Table 20 on page 51.

Figure 15: Service Collection



NOTE: Addresses must be created before you can configure a security policy. See "Address (address_collection_type)" on page 51.

Table 20: Service Collection Data Elements

Data Element	Description
service	Service rule collection.
name_	Name of the service.
service	Service type.
group	Group
global_collection	Global zone.
source	Source
type	ICMP type.
comment	Comments about the service collection.
predefined	Predefined (Boolean)
avail	Predefined service is available.
category	Category of service type.
idpname	IDP name
fwname	Firewall name
support-on_collection	Supported on collection.
versions	Service collection version.

Address (address_collection_type)

The address collection (address_collection_type) enables you to work with addresses. Addresses are the workstations, routers, switches, subnetworks, and other components that are connected to your network. In multicast routing, a multicast address specifies the multicast group to which the data is sent. Related addresses may be aggregated into address groups.

These data elements are illustrated and described in Figure 16 on page 52 and Table 21 on page 52.

Figure 16: Address Collection

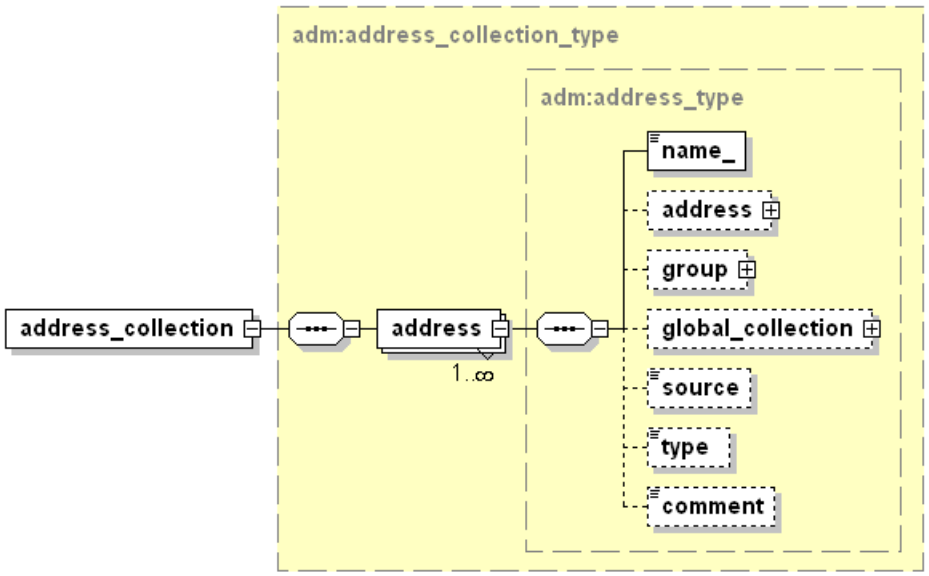


Table 21: Address Collection Data Elements

Data Element	Description
name_	Name of the device or network component.
group	Name of the group.
global_collection	Global rules collection.
source	Source of the address type.
type	Address type (host, network, or group).
comment	Comments about the address collection.

Schedule Object (scheduleobj_collection_type)

The schedule object collection (scheduleobj_collection_type) enables you to work with schedules. Schedules define a time range during which a security policy rule is in effect.

These data elements are illustrated and described in Figure 17 on page 53 andTable 22 on page 53.

Figure 17: Schedule Object

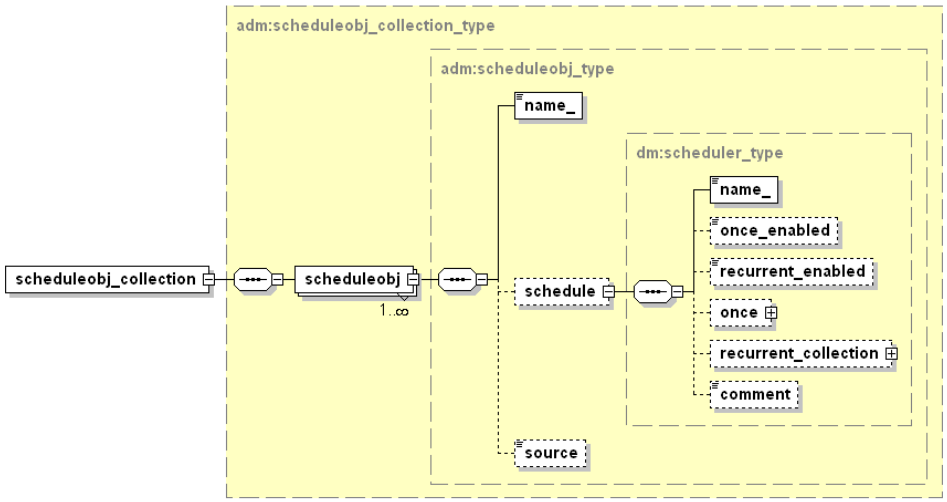


Table 22: Schedule Object Data Elements

Data Element	Description
scheduleobj	Schedule object rules
name_	Name of the schedule object type.
schedule	Schedule
source	Source
name_	Name of the scheduler type.
once_enabled	Enabled for one session.
recurrent_enabled	Enabled for recurrent use.
once	One time schedule type.
recurrent_collection	Recurrent collection.
comment	Comments about the scheduler type.

Attack (attack_collection)

The attack collection (attack_collection) enables you to counter attacks. You can configure basic information about possible attacks such as attack object severity, external references, names, and so on. You can include additional information, including general descriptions and keywords that make it easier to locate and maintain an attack object in your security policies.

NOTE: The fields that can be edited depend on the object type, compound attack object, protocol anomaly object, and signature of the attack. The signature can provide information about the protocol and context used to perpetrate the attack, whether or not the attack is considered malicious, direction and flow of the attack, signature pattern of the attack, and the values found in the header section of the attack traffic.

These data elements are described in Table 23 on page 56.

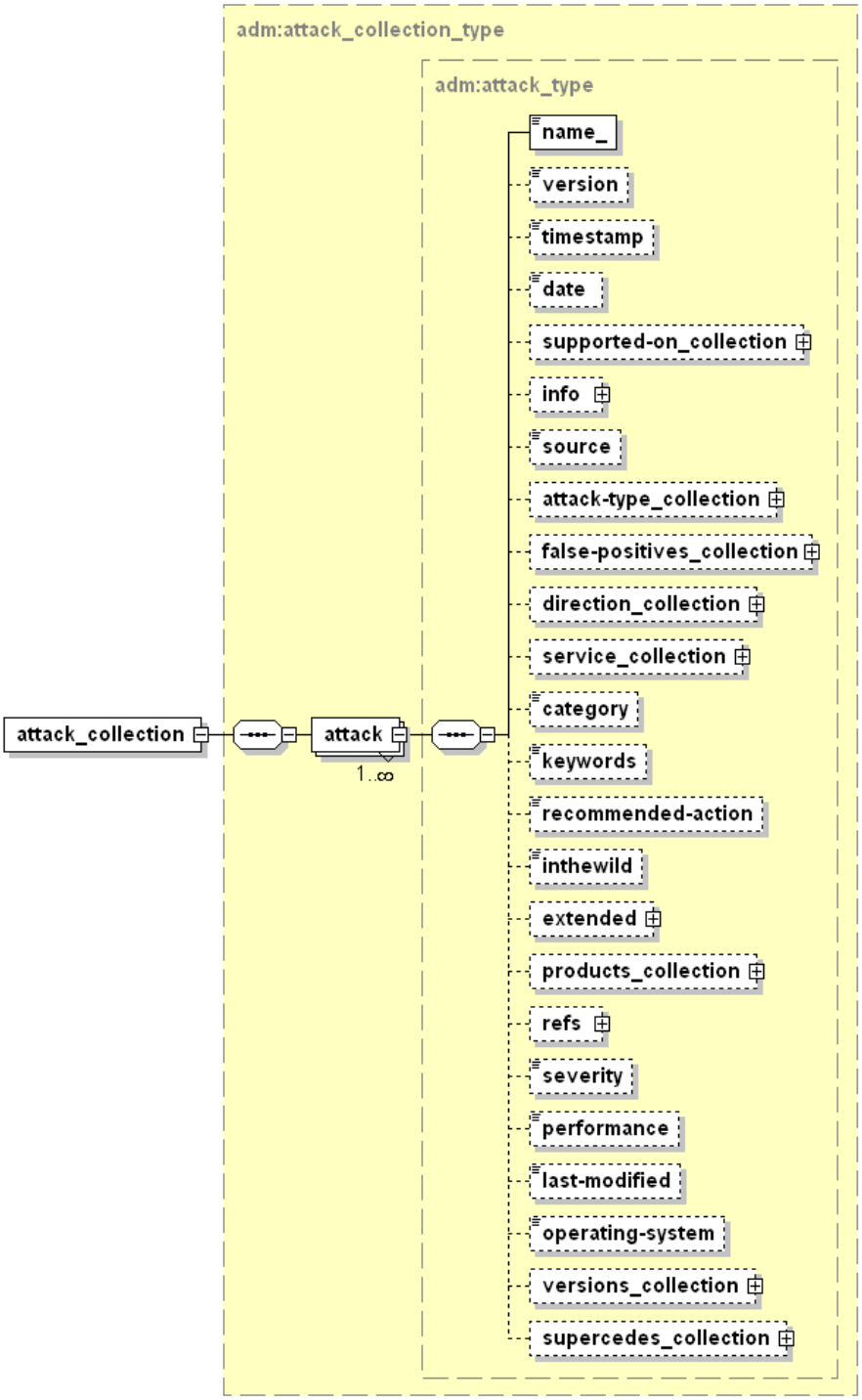


Table 23: Attack Collection Data Elements

Data Element	Description
attack	Specific type of attack.
name_	Name of the attack type.
version	NSM attack data base version.
timestamp	NSM attack database timestamp.
date	Date of the attack type.
support-on_collection	Supported on collection.
info	Information about the attack.
source	Attack type
attack-type_collection	Attack type collection.
false-positives_collection	False positives.
direction_collection	Direction collection.
service_collection	User defined services. See "Service (service_collection)" on page 50.
category	Attack category.
keywords	Keywords associated with the attack.
recommended-action	Recommended action in response to the specified type of attack. Possible values: • none (default) • ignore • drop-packet • drop • close-client • close-server • close
inthewild	Recommended
extended	Extended information.
products_collection	Products collection.
refs	References.

Table 23: Attack Collection Data Elements *(continued)*

Data Element	Description
severity	Attack severity or information about the attack. Possible values: • Critical • Major • Minor • Warning • Info
performance	Detection performance.
last-modified	Last modified.
operating-system	Operating system.
versions_collection	Version ID.
supercedes_collection	Member list

Antivirus (avobj_collection)

The Antivirus collection (avobj_collection) enables you to configure your security policies to include antivirus data. These data elements are illustrated and described in Figure 18 on page 57 and Table 24 on page 58.

Figure 18: Antivirus Collection

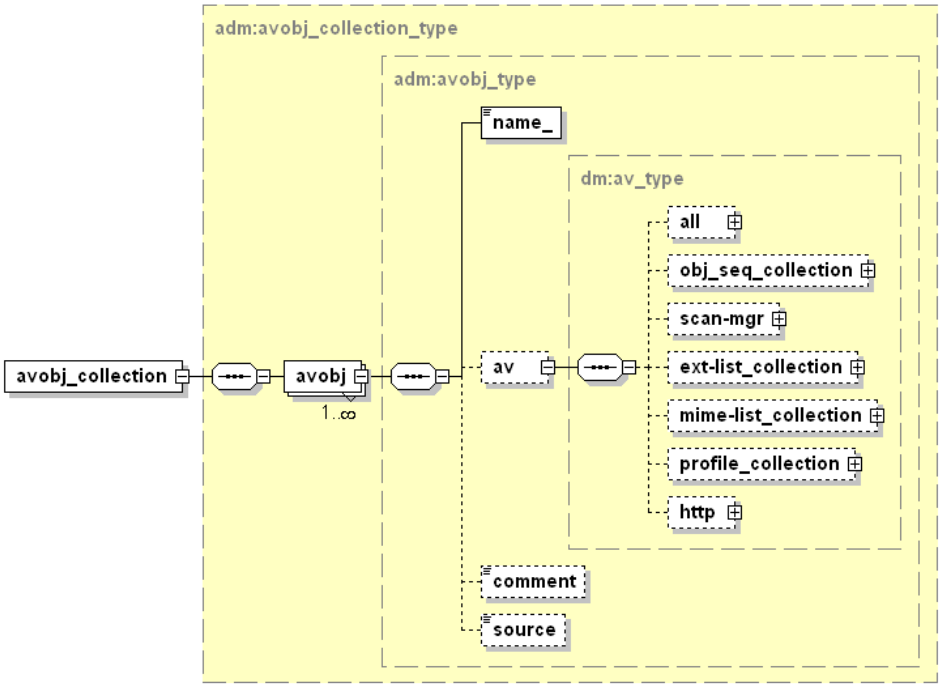


Table 24: Antivirus Collection Data Elements

Data Element	Description
avobj	Antivirus object
name_	Name of the antivirus type.
av	Antivirus type.
comment	Comments about the Antivirus type.
source	Anitvirus source
all	All antivirus types.
obj_seq_collection	All of the object sequence collection
scan-mgr	Scan manager.
ext-list_collection	File extension lists.
mime-list_collection	Mime type lists.
profile_collection	Profile
http	HTTP

GTP (gtpobj_collection_type)

The GPRS Tunneling Protocol (GTP) collection (gtp_collection) enables you to configure your security policies to handle GTP traffic. These data elements are illustrated and described in Figure 19 on page 59 and Table 25 on page 59.

Figure 19: GTP Collection

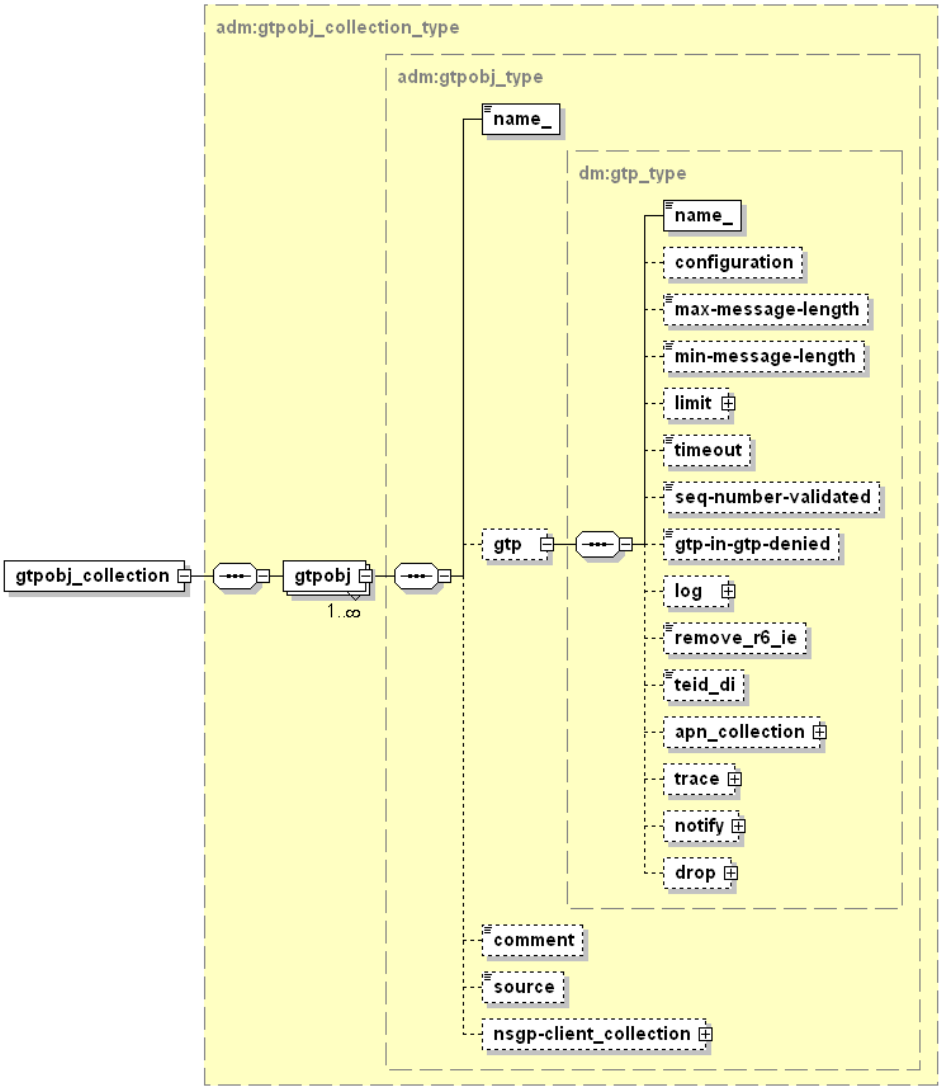


Table 25: GTP Collection Data Elements

Data Element	Description
gtpobj	GTP object
name_	Name of the GTP object type.
gtp	GTP object
comment	Comments about the GTP collection.
source	Source of the GTP object.
nsgp-client_collection	NSGP clients.

Table 25: GTP Collection Data Elements *(continued)*

Data Element	Description
name_	Name of the GTP type.
configuration	Configuration
max-message-length	Maximum message length. Default = 65535 bytes.
min-message-length	Minimum message length. Default = 0 bytes.
limit	GNS limit
timeout	Inactivity period after which a session is removed from a security device. Possible values: • never = no timeout • default = default period of time • user-defined = user defined inactivity timeout period in minutes.
seq-number-validated	Sequence number validation
gtp-in-gtp-denied	GTP in GTP denied
log	GTP logging.
remove_r6_ie	Not used often.
teid_di	Not used often.
apn_collection	IMSI prefix and APN filtering
trace	Subscriber trace
notify	NSGP notification
drop	GTP message content filtering.

DI Profile (DIProfile_collection_type)

A Deep Inspection (DI) Profile collection contains predefined attack object groups (supplied by Juniper Networks) and your own custom attack object groups.

These data elements are illustrated and described in Figure 20 on page 61 and Table 26 on page 61.

Figure 20: DI Profile

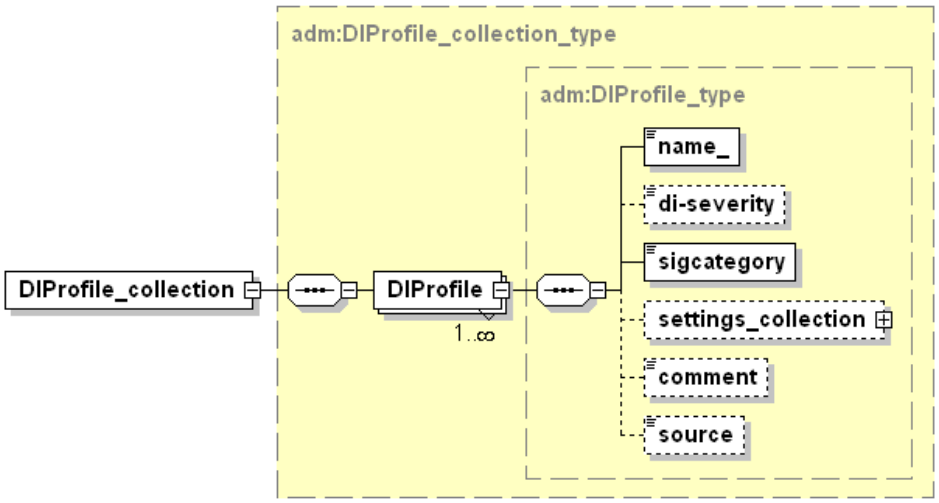


Table 26: DIP Data Elements

Data Element	Description
DIProfile	DI Profile collection.
name_	Name of the DI Profile type.
di-severity	DI Severity
sigcategory	Sign category associated with the DIProfile type.
settings_collection	Profile settings.
comment	Comments about the DIP collection.
source	DI Profile

Global DIP (globaldip_collection)

The Global Dynamic IP (DIP) collection (globaldip_collection) data elements represent various global DIP settings in a security policy.

These data elements are illustrated and described in Figure 21 on page 62 and Table 27 on page 62.

Figure 21: Global DIP Collection

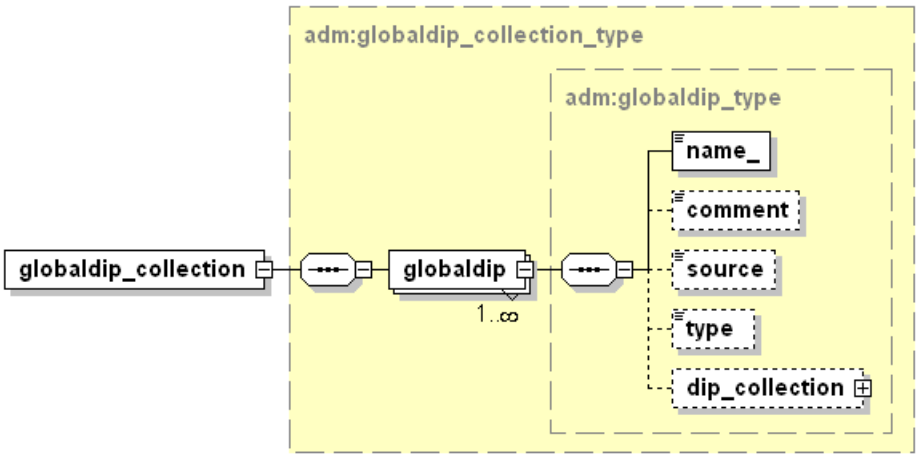


Table 27: Global DIP Data Elements

Data Element	Description
globaldip	Global DIP collection
name_	Name of the global DIP type.
comment	Comments about the Global DIP collection.
source	Global DIP source type
type	Type of Global DIP
dip_collection	Deep inspection profile collection

Global MIP (globalmpi_collection)

The Global Mapping IP (MIP) collection (globablmpi_collection) data elements represent various mapped IP (MIP) settings in a security policy.

These data elements are illustrated and described in Figure 22 on page 63 and Table 28 on page 63.

Figure 22: Global MIP Collection

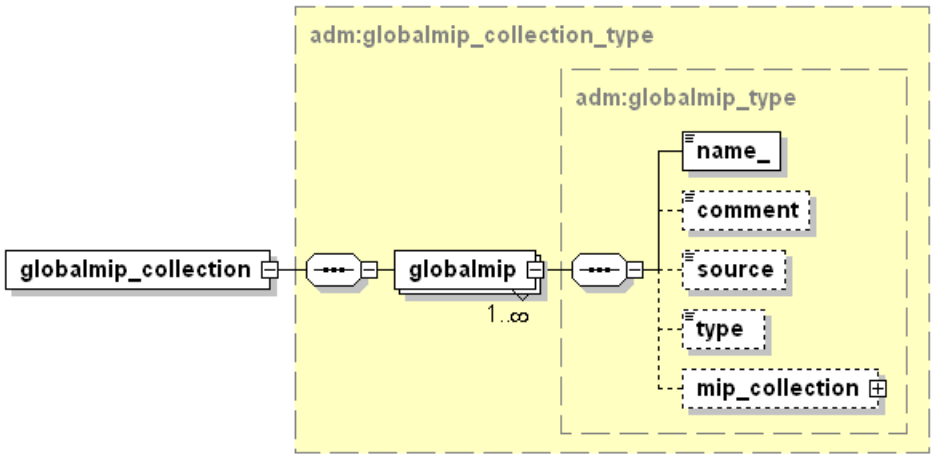


Table 28: Global MIP Data Elements

Data Element	Description
globalmip	Global MIP collection type
name_	Name of the global MIP type.
comment	Comments about the global MIP collection.
source	Global MIP source type
type	Type of Global MIP
mip_collection	MIP

Global VIP (globalvip_collection)

The Global VIP collection (globalvip_collection) data elements represent various global virtual IP (VIP) settings in a security policy.

These data elements are illustrated and described in Figure 23 on page 64 and Table 29 on page 64.

Figure 23: Global VIP Collection

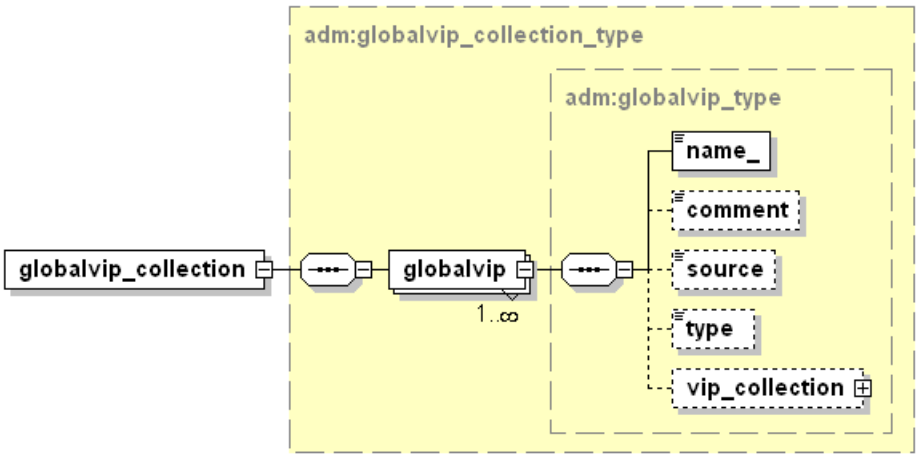


Table 29: Global VIP Data Elements

Data Element	Description
globalvip	Global VIP collection
name_	Name of the global VIP collection type.
comment	Comments about global VIP collection.
source	Global VIP source type
type	Type of Global VIP
vip_collection	VIP

URL Filter Object (urlfilter_collection)

The URL Filter Object collection (urlfilter_collection) data elements represent various URL filter object settings in a security policy.

These data elements are illustrated and described in Figure 24 on page 65 and Table 30 on page 65.

Figure 24: URL Filter Object Collection

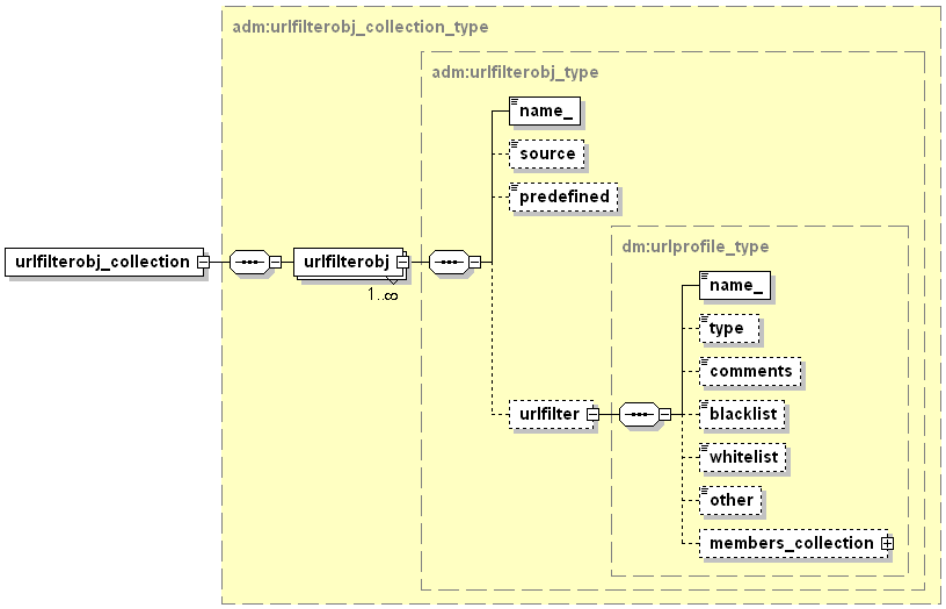


Table 30: URL Filter Data Collection

Data Element	Description
urlfilterobj	URL filter object collection.
name_	Name of the URL filter object type
source	Predefined web filter source
predefined	Predefined Web profile
urlfilter	Web profile
name_	Name of the URL profile type.
type	Type of URL filter object.
comments	Comments about the URL filter collection.
blacklist	Blacklisted URL (sites denied)
whitelist	Whitelisted URLs (sites permitted)
other	Action for all other URLs
members_collection	Members categories

NAT-DIP (dip_collection)

The NAT-DIP Object collection (dip_collection) data elements represent various NAT-DIP settings in deviceobj configuration. These data elements are illustrated in Figure 25 on page 66 and described in Table 31 on page 66.

Figure 25: NAT- DIP Object Collection

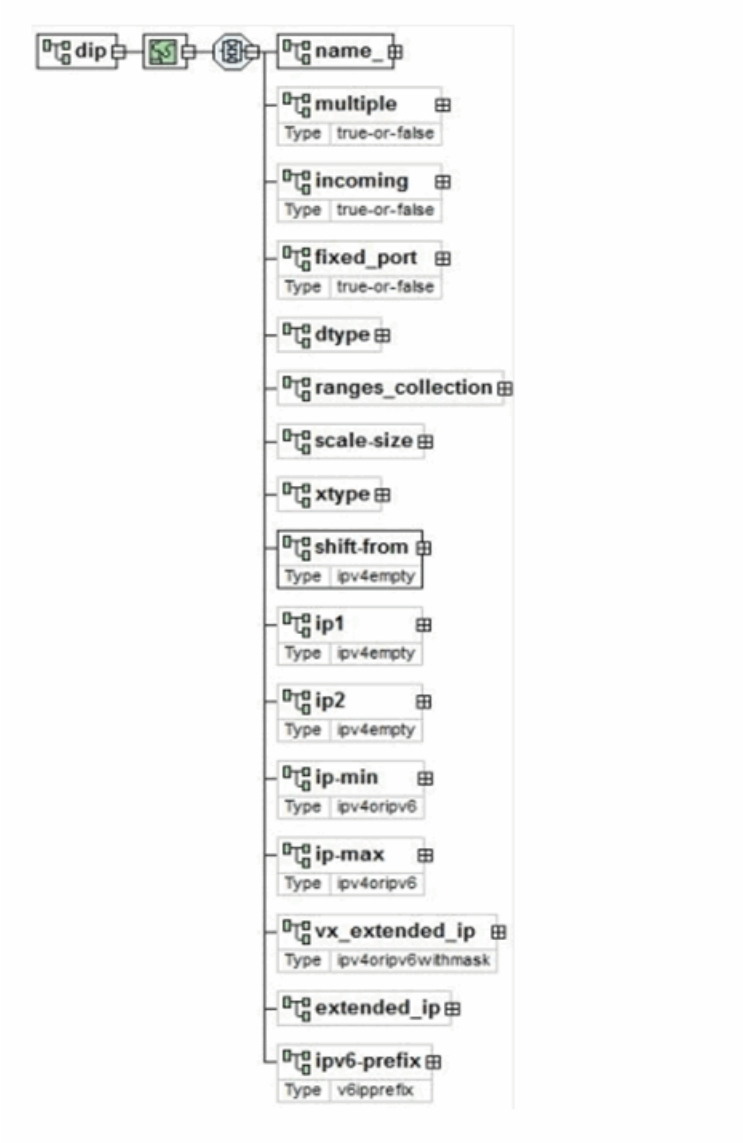


Table 31: NAT DIP Data Elements

Data Element	Description
dip	DIP collection

Table 31: NAT DIP Data Elements *(continued)*

Data Element	Description
name_	ID of the DIP object
multiple	Multiple DIP ranges used
incoming	Incoming NAT
Fixed_port	Fixed port
dtype	Type of data, that is, ipv4/ipv6
Ranges_collection	Range of IPs
xtype	Extended IP type
Ip1	Lower IP
Ip2	Upper IP
Ip-max	Upper IP of type ipv6 or ipv4
Ip-min	Lower IP of type ipv6 or ipv4
Vx_extended_ip	Extended IP
Extended_ip	Extended IP collection
Ipv6-prefix	IPV6 prefix value

NAT-MIP (Mip_collection)

The NAT-MIP Object collection (mip_collection) data elements represent various NAT-MIP settings in deviceobj configuration. These data elements are illustrated in Figure 26 on page 68 and described in Table 32 on page 68.

Figure 26: NAT- MIP Object Collection

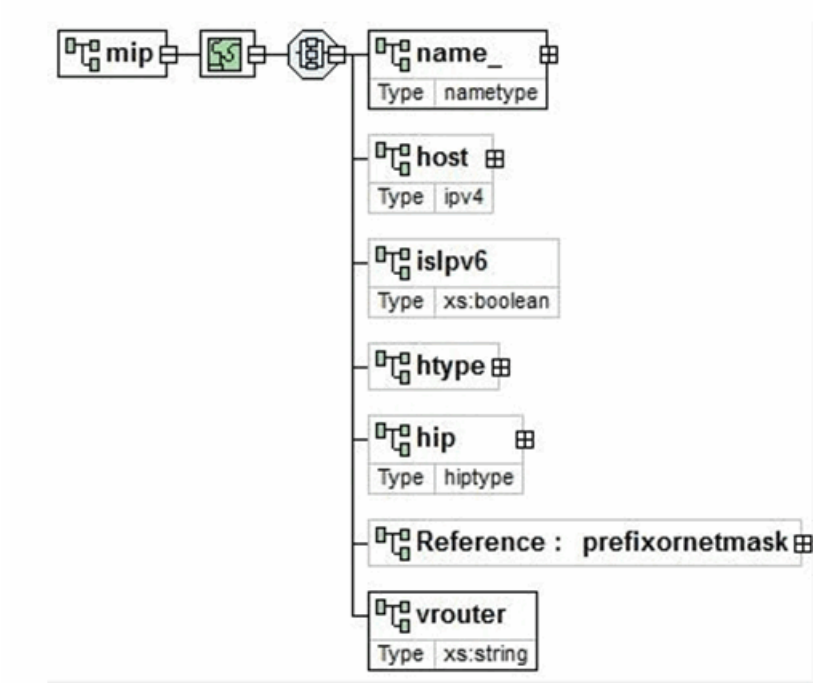


Table 32: NAT-MIP Data Elements

Data Element	Description
mip	NAT-MIP objects
name_	ID of the MIP object
host	IP of the host
isIpv6	Ipv6 enabled flag
htype	Host IP type
hip	Host
Prefix/netmask	Netmask value
vrouter	Host virtual router name

NAT-VIP (vipSet)

The VIP set data elements represent various NAT-VIP settings in deviceobj configuration. These data elements are illustrated in Figure 27 on page 69 and described in Table 33 on page 69.

Figure 27: NAT- VIP Object Collection

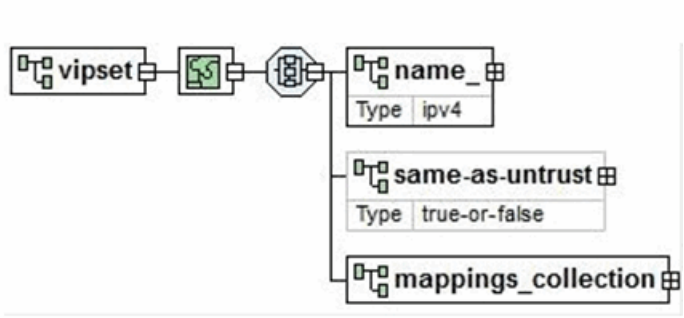


Table 33: NAT-VIP Data Elements

Data Element	Description
vipset	VIP object
name_	ID of the VIP object
Same-as-untrust	Same as Interface IP
Mappings-collection	Port mapping collection

Static Route

The Static Route data elements represent various static route settings in deviceobj configuration. These data elements are illustrated in Figure 28 on page 70 and described in Table 34 on page 70.

Figure 28: Static Route Object Collection

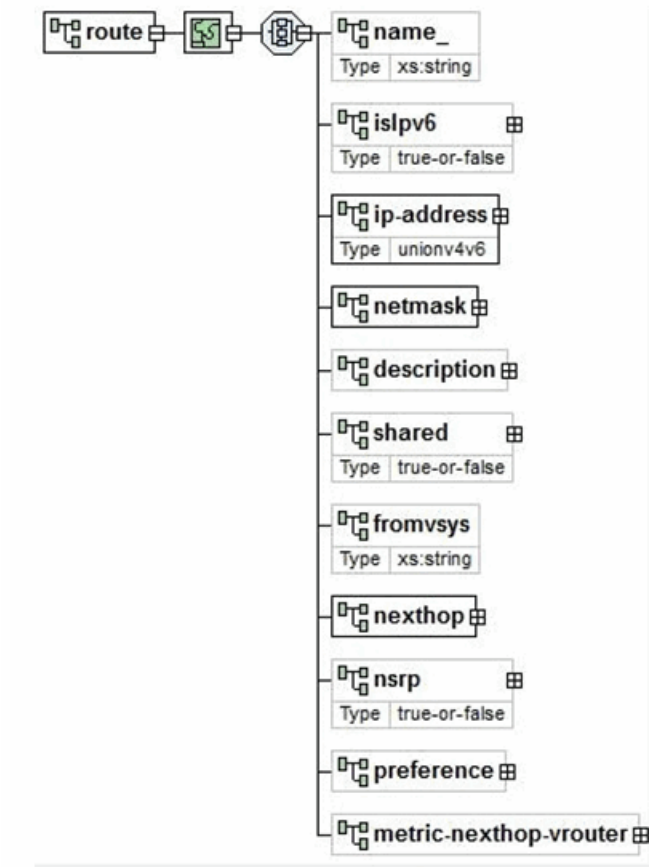


Table 34: Static Route Data Elements

Data Element	Description
route	Route list
name_	Route node name
islpv6	Ipv6 enabled or not
Ip-address	IP address
netmask	network mask
From vsys	Route is from vsys
nexthop	Next hop address

Gateway

The Gateway data elements represent various gateway settings in deviceobj-vpn settings configuration. These data elements are illustrated in Figure 29 on page 71 and described in Table 35 on page 71.

Figure 29: Gateway Object Collection

Table 35: Gateway Data Elements

Data Element	Description
peer	Gateway object
name_	Gateway name
Peer-info	Remote gateway type

Table 35: Gateway Data Elements *(continued)*

Data Element	Description
version	ikev1 or ikev2
dpd	Dead Peer Detection
mode	Mode (Initiator)
Peer-id	Peer ID.
Local-id	Local ID
Outgoing-ifp	Outgoing interface
Outgoing-zone	Outgoing zone
Local-address	Local address
Preshare_rsa_dsa	Preshare Key/RSA/DSA
proposal	Security level
heartbeat	Heartbeat
natt	NAT traversal
xauth	XAuth config collection
modecfg	Modecfg collection
accounting	Accounting server details

VPN (vpn_collection)

The VPN (vpn_collection) data elements represent various vpn settings in deviceobj configuration. These data elements are illustrated in Figure 30 on page 73 and described in Table 36 on page 73.

Figure 30: VPN Object Collection

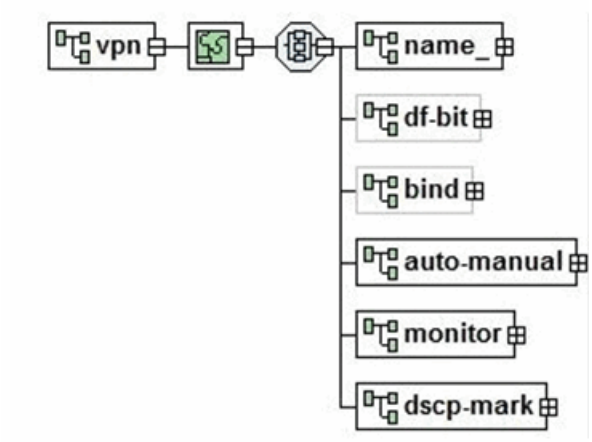


Table 36: VPN Data Elements

Data Elements	Description
vpn	VPN object collection
name_	VPN name
Df-bit	Not to set fragment bit in the header
bind	Bind tunnel interface
Auto-manual	AutoKey IKE/Manual Key VPN collection
monitor	VPN monitor
Dscp-mark	DSCP marking

USER (user_collection)

The User (user_collection) data elements represent various admin user settings in deviceobj configuration. These data elements are illustrated in Figure 31 on page 74 and described in Table 37 on page 74.

Figure 31: USER Object Collection

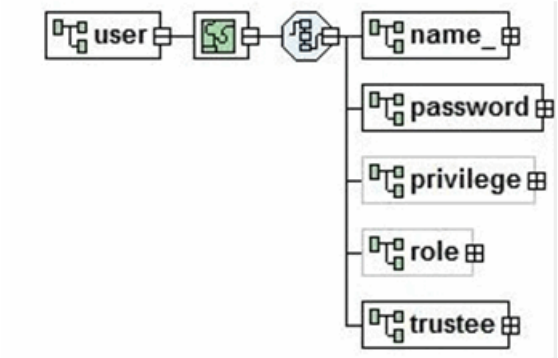


Table 37: USER Data Elements

Data Element	Description
user	User collection
name_	Admin user name
password	Password of the admin user
privilege	Permission to the user. For example: Root, read-only, and all
role	Role of the user. For example: audit, crypto, and security
trustee	Trustee, that is, interface and modem nodes

PART 3

Using the NSM API from a Perl Client

This chapter explains how to install the Perl Client environment, use the client to access the NSM API, and use the API to manage shared objects.

- Installing the Perl Client Environment on page 77
- Using the Perl Client to Access the NSM API on page 83
- Using the API to Manage Shared Objects on page 85

CHAPTER 6

Installing the Perl Client Environment

This section explains how to install the Perl Client environment.

- Installing the Perl Client Environment on Linux-Unix Machines on page 77
- Installing the Perl Client Environment on Windows Machines on page 78
- NSM Perl API Directory Structure on page 79
- Using a Perl Script to Access the NSM API on page 79

Installing the Perl Client Environment on Linux-Unix Machines

Before you install this client environment, install the following software on your machine:

- Perl version 5.14.2

  Perl is available (free) at http://www.activestate.com/Products/activeperl/index.mhtml.

  Perl is included as standard package in all Unix and Linux Systems.
- openssl installed under /usr.

To install the client environment on a Linux-Unix machine:

1. Launch cpan as root. Do one of the following:
   - Execute the cpan program:

     cpan
   - Run the perl -MCPAN -e shell:

     perl -MCPAN -e
2. Update cpan, accepting all defaults: **cpan[1]> install cpan**
3. Install the CPAN bundle: **cpan[1]> install Bundle::CPAN**
4. Install Crypt::SSLeay: **cpan[1]> install Crypt::SSLeay** .
5. Install LWP: **cpan[1]> install LWP**
6. install XML Simple: **cpan[1]> install XML::Simple**
7. Install MIME Tools: **cpan[1]> install MIME::Tools**
8. Install the MIME Parser: **cpan[1]> install MIME::Parser**

9. Install Data Dumper Module: **cpan[1]> install Data::Dumper**
10. Install SOAP Lite (do not accept the default): **cpan[1]> install SOAP::Lite**
11. Enable https, MIME, DIME, and Axis2 MIME support.

NOTE: All the Perl module installation listed above should be successful. If you are using CPAN, dependencies are installed when the package is installed.

Installing the Perl Client Environment on Windows Machines

Before you install this client environment, install the following software on your machine:

- Perl version 5.14.2

  Perl is available (free) at http://www.activestate.com/Products/activeperl/index.mhtml.

  Alternatively, you can download Strawberry Perl available at http://strawberryperl.com/. This package has built-in Perl modules specifically used for Windows Platform.

- openssl installed under C:\

  Openssl is available (free) at http://www.slproweb.com/products/Win32OpenSSL.html.

To install the client environment on a Windows machine:

1. In the Windows shell, launch cpan. Do one of the following:
   - Execute the cpan program:

     cpan

   - Run the Perl -MCPAN -e shell:

     perl -MCPAN -e
2. Update cpan, accepting all defaults: **cpan[1]> install cpan**
3. Install YAML: **install YAML**.
4. Install the CPAN bundle: **cpan[1]> install Bundle::CPAN**
5. Install Crypt::SSLeay: **cpan[1]> install Crypt::SSLeay**
6. Install LWP: **cpan[1]> install LWP**
7. install XML Simple: **cpan[1]> install XML::Simple**
8. Install MIME Tools: **cpan[1]> install MIME::Tools**
9. Install the MIME Parser: **cpan[1]> install MIME::Parser**
10. Install Data Dumper Module: **cpan[1]> install Data::Dumper**
11. Install SOAP Lite (do not accept the default): **cpan[1]> install SOAP::Lite**
12. Enable https, MIME, DIME, and Axis2 MIME support.
13. Upgrade all modules, accepting the defaults: **cpan> upgrade**

The upgrade process takes a few minutes.

NOTE: All the above Perl module installation should be successful. If you are using CPAN, dependencies are installed when the package is installed.

NSM Perl API Directory Structure

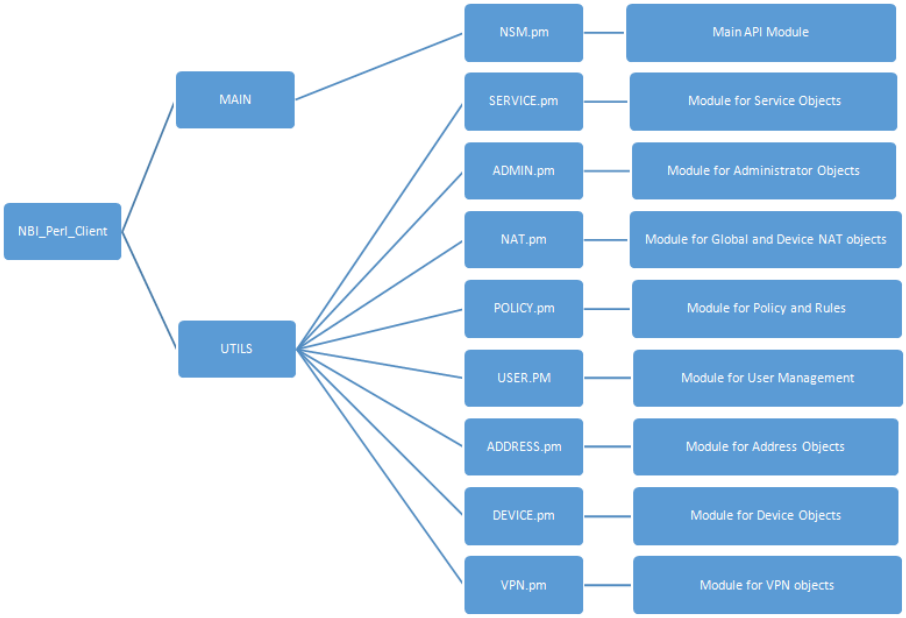
You can find the Perl API scripts in the following folder in all NSM builds.

Directory: **/usr/netscreen/GuiSvr/lib/webproxy/client**.

Filename: **NBI_Perl_Client.tar**.

After you untar the **NBI_Perl_Client.tar** file, the files are extracted to the directory structure as illustrated in Figure 32 on page 79. The figure also provides the module name and functionality information.

Figure 32: NSM Perl API Directory Structure



Using a Perl Script to Access the NSM API

This example shows how to log in to the server and retrieve system information for the server.

Sample Perl Login Script

```
#!/usr/bin/perl -w

use SOAP::Lite +trace => 'all';
```

```
#use SOAP::Lite;
delete $ENV{'https_proxy'};

use constant NS_XSD=> 'http://www.w3.org/1999/XMLSchema'; #XSD 1999 schema

use constant NS_XSI=> 'http://www.w3.org/1999/XMLSchema-instance'; #XSI 1999
schema

our @NSM_SERVERS = qw(
      10.157.39.201:8443
      8443
);

our $LOGIN_TOKEN;
sub soap_call($$$) {
  my $service = shift;
  my $method = shift;
  my $input = shift;

  my $nbi_method = "https://";
  my $nbi_uri = "/axis2/services";
  my $jp_url = "http://juniper.net/webproxy";

  if ( !defined $main::ACTIVE_SERVER ) {
      if ( scalar @main::NSM_SERVERS == 0 ) {
            print ("Couldn't connect to any NSM Servers", "\n");
            exit 1;
      } else {
            $main::ACTIVE_SERVER = shift @main::NSM_SERVERS;
      }
  }


my $nsm_url = qq|${nbi_method}${main::ACTIVE_SERVER}${nbi_uri}|;
  my $soap_service = SOAP::Lite
      -> proxy ("$nsm_url/$service")
      -> uri  ( "$jp_url/" . lc($service) )
      -> on_fault (
            sub {
                  if ( $_[0]->transport->status =~ /^503/ ) {
                        undef $main::ACTIVE_SERVER;
                        soap_call($service,$method,$input);
                  }
            }
      ) ;


my $soap_method = SOAP::Data->name($method)->attr( {'xmlns' => "$jp_url/" .
lc($service),'xmlns:xsd'=>NS_XSD, 'xmlns:xsi'=>NS_XSI} );
  # Execute and grab response
  my $response;

  if ( defined $input ) {
      $response = $soap_service->call($soap_method, @{$input});
  } else {
      $response = $soap_service->call($soap_method);
  }

  if ($response->fault) {
        print "$service#$method: " . $response->faultcode .": " .
$response->faultstring . "\n";
```

```
        exit 1;
  }

 return $response;
}

sub login($$) {
  my $username = shift;
  my $password = shift;

  my @apiLogin = (
      SOAP::Data->name('domainName')->value('global'),
      SOAP::Data->name('userName')->value($username),
      SOAP::Data->name('password')->value($password)
 );

  my $response = soap_call("SystemService","LoginRequest",\@apiLogin);
  my $loginStatus = $response->valueof("//Body/LoginResponse/loginStatus/status");

  if ($loginStatus eq "Success" ) {
      $main::LOGIN_TOKEN =
$response->valueof("//Body/LoginResponse/authToken/Token");
  } elsif ( $loginStatus eq "Failure" ) {
      print "Invalid credentials", "\n";
      exit 1;
 }
}


sub get_all_sds() {
  my @token = (
     SOAP::Data->name('Token')->value($LOGIN_TOKEN)->prefix('ns1')
     );
  my @AuthToken = (

SOAP::Data->name('AuthToken')->value(\@token)->prefix('ns1')->uri('http://juniper.net/core')

     );
  my $response = soap_call("SystemService","GetSystemInfoRequest", \@AuthToken);


  return $response;
}

login("super","netscreen");
print $LOGIN_TOKEN, "\n";
get_all_sds;
```

NOTE: If you are using NSMXpress, the API client must connect to the TCP Port 443.

CHAPTER 7

Using the Perl Client to Access the NSM API

This section explains how to use the Perl Client library for NSM to access the NSM API. The library is located in the directory **$NSROOT/GuiVar/webproxy/clienton** NSM server.

- Login and Logout on page 83

Login and Logout

Enter the following commands to log into and log out of the Perl Client Library.

```
# Login:

  use SOAP::Lite;
  use MAIN::NSM;


$host = '<IP ADDRESS>';
$USERNAME = '<USERNAME>';
$PASSWORD = '<PASSWORD>';

$connect = MAIN::NSM->new( 'HOST' => "$host", 'PORT' => "$port", );

my $connect_info = $connect->login( 'UNAME' => $USERNAME, 'PASSWD' => $PASSWORD
);


#Log out:

  $connect->logout;
```

CHAPTER 8

Using the API to Manage Shared Objects

The Perl client has three object modules: Address (read/write/delete), Service (read/write/delete), and Device (read only). The Address module is used to initialize, add (host, network, multicast, group, and global), replace, rename, read, and delete address objects. The Service module is used to initialize, add (group and global), and delete service objects. The Device module is used to initialize and read device objects.

This section explains how to use the modules and write programs.

- Using the Perl Client Library with Address Objects on page 85
- Using the Perl Client Library with Service Objects on page 89
- Using the Perl Client Library with Device Objects on page 93
- Troubleshooting Common Perl API issues on page 94

Using the Perl Client Library with Address Objects

You can use the Perl client library to add, replace, rename, read, and delete address objects. These tasks are summarized below.

NOTE: The COLOR, COMMENT, DOMAIN, and DEVICE (any/any is the default) arguments are optional for these procedures.

Add Address Objects

This section shows how to add an address object.

To add address objects:

1. Log in to the Perl client.
2. Initialize the address object.

   Enter:

   ```
   my $address = UTILS::ADDRESS->init( 'SOAP'=>$connect );
   ```

3. Add the host, network, multicast, group, and global objects.

NOTE:

- <OBJECT_NAME> – Name of the Object that is added to NSM DB.
- Use NSM Login and Logout function as defined above.

- **Add a host with IP/Mask**

  Enter:

```
$result  = $address->addHostObjects('OBJECT_NAME'=>'Host_IP-'<OBJECT_NAME>,
 'IP'=>'200.200.20.0', 'MASK'=>'30');
   put_log('WARN', msg=>"Error while adding Host objects." ) unless( $result
 );
```

- **Add a host with domain**

  Enter:

```
$result  = $client->addHostObjects('OBJECT_NAME'=>'Host_Domain-'.
<OBJECT_NAME>);
put_log('WARN', msg=>"Error while adding Host objects." ) unless( $result
);

$result  = $address->addHostObjects('OBJECT_NAME'=>'Foo-'.
<OBJECT_NAME>, 'DOMAIN'=>'englab.juniper.net' );
put_log('WARN', msg=>"Error while adding Host objects." ) unless( $result
);
```

- **Add the network**

  Enter:

```
  $result  = $address->addNetworkObjects('OBJECT_NAME'=>'Network-'.
<OBJECT_NAME>, 'IP'=>'100.100.10.0', 'MASK'=>'30', 'COLOR'=>'blue' );
    put_log('WARN', msg=>"Error while adding Network objects." ) unless(
$result );
```

- Add multicast

  Enter:

```
$result  = $address->addMulticastObjects('OBJECT_NAME'=>'Multicast-'.
<OBJECT_NAME>, 'IP'=>'230.196.0.0', 'MASK'=>'16', 'COLOR'=>'cyan' );
    put_log('WARN', msg=>"Error while adding Multicast objects." ) unless(
$result );
```

- Add group objects

  Enter:

```
@members = ('Foo-3','Foo-4');
    $result  = $address->addGroupObjects('OBJECT_NAME'=>'New-Group-2',
  'GROUP_MEMBERS'=>\@members );
put_log('WARN', msg=>"Error while adding global objects." ) unless( $result
 );

  @members = ( {  'DOMAIN'=>'any',
                  'DEVICE'=>'any',
                  'ADDRESS'=>'Foo-5'
                    },
```

```
                    {  'DOMAIN'=>'global',
                       'DEVICE'=>'<DEVICE NAME>',
                       'ADDRESS'=>'Foo-6'
                         } );
```

- Add global objects

  Enter:

```
$result  = $address->addGlobalObjects('OBJECT_NAME'=>'Poly-1',
'GROUP_MEMBERS'=>\@members  );

  @members = ( {  'DOMAIN'=>'any',
                  'DEVICE'=>'any',
                  'ADDRESS'=>'Foo-7'
                    } );
  $result  = $address->addGlobalObjects('OBJECT_NAME'=>'Poly-2',
'GROUP_MEMBERS'=>\@members  );

  @members = ( { 'ADDRESS'=>'Foo-8' } );
  $result  = $address->addGlobalObjects('OBJECT_NAME'=>'Poly-3',
'GROUP_MEMBERS'=>\@members  );
```

4. Log out.

Replace an Address Object

This section shows how to replace an existing address objects.

To replace an address object:

Follow these steps to replace an existing object.

1. Log into the Perl client.
2. Replace the object.

   Enter:

```
$result  = $address->replaceHostObjects('OBJECT_NAME'=>'Foo-'.
<OBJECT_NAME>, 'DOMAIN'=>'spglab.juniper.net', 'COLOR'=>'green' );
   put_log('WARN', msg=>"Error while replacing Host objects." ) unless( $result
 );
my @members = ('Foo-7','Foo-8');
$result   = $address->replaceGroupObjects('OBJECT_NAME'=>'New-Group-1',
'GROUP_MEMBERS'=>\@members, 'COLOR'=>'red' );

  put_log('WARN', msg=>"Error while adding group objects." ) unless( $result
);
```

3. Log out.

Rename Address Objects

This section shows how to rename an existing address object.

To rename an address objects:

1. Log into the Perl client.

2. Rename the object.

   Enter:

```
for( my $i = 1; $i < 10; $i++ ){

    $result  = $address->renameObjects('OBJECT_NAME'=>'Network-'.$i,
'NEW_NAME'=>'MyNetwork-'.$i);                    put_log('WARN', msg=>"Error
 while renaming Network objects." ) unless( $result );
          }
```

3. Log out.

Read Address Objects

This section shows how to read address objects.

To read address objects:

1. Log into the Perl client.
2. Read the object.

   Enter:

   Get Host Objects:

```
my ( $result, $values ) ;
                                        ( $result, $values )  =
$client->getHostObjects('OBJECT_NAME'=>['Foo-1','Foo-2','Foo-3','Foo-4']);
                 put_log('WARN', msg=>"Error while reading Host objects." )
unless( $result );

  while( my $hash = shift(@{$values} ) ){

    put_log('INFO', msg=>"Result ==> $hash->{'OBJECT_ID'} :
$hash->{'OBJECT_NAME'} : $hash->{'DOMAIN'} : $hash->{'ZONE'}");

  }
```

   Get Group Objects:

```
( $result, $values )  =
$client->getGroupObjects('OBJECT_NAME'=>['New-Group-1','New-Group-2']);
               put_log('WARN', msg=>"Error while reading Group objects." )
unless( $result );
while( my $hash = shift(@{$values} ) ){
                                            put_log('INFO', msg=>"Result ==>
$hash->{'OBJECT_ID'} : $hash->{'OBJECT_NAME'}");
      while( my $tmp = shift(@{$hash->{'MEMBERS'}} ) ){
                                                 put_log('INFO',msg=> "Result
--> $tmp\n" );
        }
  }
```

   Get Global Objects:

```
( $result, $values )  =
$client->getGlobalObjects('OBJECT_NAME'=>['Poly-1','Poly-2','Poly-3']);
                 put_log('WARN', msg=>"Error while reading Global objects." )
 unless( $result );
```

```
                while( my $hash = shift(@{$values} ) ){
                                                        put_log('INFO',
msg=>"Result ==> $hash->{'OBJECT_ID'} : $hash->{'OBJECT_NAME'}");
                        while( my $hashref = shift(@{$hash->{'MEMBERS'}} ) ){

put_log('INFO',msg=> "Result -->
".$hashref->{'DOMAIN'}.":".$hashref->{'DEVICE'}.":".$hashref->{'ADDRESS'} );
    }
  }
```

3. Log out.

Delete Address Objects

This section shows how to delete specific address objects.

To delete address objects:

1. Log into the Perl client.
2. Delete the object.

   Enter:

```
 $result  = $client->deleteObjects('OBJECT_NAME'=>
['Multicast-1','Multicast-2','Multicast-3','Multicast-4','Multicast-5','Multicast-6']
 );
put_log('WARN', msg=>"Error while deleting Multicast objects." ) unless( $result
 );
```

3. Log out.

Delete All Address Objects

This section shows how to delete all address objects.

To delete all address objects:

1. Log into the Perl client.
2. Delete all address objects.

   Enter:

```
$result  = $client->deleteAllObjects();
                                puput_log('WARN', msg=>"Error while deleting
all objects." ) unless( $result );
```

3. Log out.

Using the Perl Client Library with Service Objects

You can use the Perl Client Library to add, read, replace, and delete service objects. These activities are summarized in the following sections

NOTE: The COLOR and COMMENT arguments are optional for these procedures.

Add Service Objects

This section shows how to use the Perl Client Library to add service objects.

To add a service object:

1. Log in to the Perl client.
2. Initialize the service object.

```
my $service = UTILS::SERVICE->init( 'SOAP'=>$connect );
```

3. Add the object.

   Enter:

   ADD ICMP TYPE

```
   'IS_ICMP'=>[
                                                {'TYPE'=>'10', 'CODE'=>'11'},

         {'TYPE'=>'20', 'CODE'=>'21'}
                                                 ]

            };

my $result  = $service->addServiceObjects('OBJECT_NAME'=>'Service-1',
'APPLICATION'=>'FTP', 'GROUP_MEMBERS'=>$members  );
```

   ADD NON ICMP TYPE

```
$members = {
                                          'NOT_ICMP'=>[

                {'PROTOCOL'=>'UDP'},

{'PROTOCOL'=>'TCP','SRC_TYPE'=>'specific','DST_TYPE'=>'specific'},

{'PROTOCOL'=>'IGMP','SRC_TYPE'=>'specific-string','DST_TYPE'=>'specific-string'},

{'PROTOCOL'=>'IP','SRC_TYPE'=>'range','DST_TYPE'=>'range'}
                                         ]};
                                                                  $result  =
$service->addServiceObjects('OBJECT_NAME'=>'Service-2', 'APPLICATION'=>'DNS',
 'GROUP_MEMBERS'=>$members  );
```

   SUN RPC TYPE

```
 $members = {
                                          'SUN_RPC'=>[

                {'SUN_LOW'=>'101'},
                                                        {'SUN_LOW'=>'201'}

                 ]};
                                          $result  =
$service->addServiceObjects('OBJECT_NAME'=>'Service-3', 'APPLICATION'=>'DNS',
 'GROUP_MEMBERS'=>$members  );
```

MS-RPC TYPE

```
$members = {
                                        'MS_RPC'=>[

              {},{}
                                                    ]};


  $result  = $service->addServiceObjects('OBJECT_NAME'=>'Service-4',
'APPLICATION'=>'DNS', 'GROUP_MEMBERS'=>$members  );
```

4. Log out.

Add Group-Global Service Objects

This section shows how to add Group-Global service objects.

To add a group/global service object:

1. Log into the Perl client.
2. Add the object.

   Enter:

   ADD GROUP SERVICE

```
 my $result  = $service->addGroupObjects( 'OBJECT_NAME'=>'Group-1' );

my $members = ['EGP'];
                                        $result  = $service->addGroupObjects(
'OBJECT_NAME'=>'Group-2',  GROUP_MEMBERS'=>$members );

$members = ['BGP','EGP'];
                                        $result  = $service->addGroupObjects(
'OBJECT_NAME'=>'Group-3',  'GROUP_MEMBERS'=>$members );
```

   ADD GLOBAL SERVICE

```
$result  = $service->addGlobalObjects( 'OBJECT_NAME'=>'Poly-1' );
                                        $members = [{'DOMAIN'=>'global'}];

 $result  = $service->addGlobalObjects( 'OBJECT_NAME'=>'Poly-2',
'GROUP_MEMBERS'=>$members );                        $members =
[{'DEVICE'=>'droopy'},{'SERVICE'=>'EGP'}];
                            $result  = $service->addGlobalObjects(
'OBJECT_NAME'=>'Poly-3',  'GROUP_MEMBERS'=>$members );
```

```
$members = [{},{}];
                                          $result  = $service->addGlobalObjects(
 'OBJECT_NAME'=>'Poly-4',  'GROUP_MEMBERS'=>$members );
```

3. Log out.

Read Group-Global Service Objects

This section shows how to use the Perl Client Library to read Group-Global service objects.

To read a service object:

1. Log into the Perl client.
2. Read the object.

   Enter:

   GET SERVICE

```
my ( $result, $values ) ;


  ( $result, $values )  =
$service->getServiceObjects('OBJECT_NAME'=>['Service-1','Service-2','Service-3','Service-4','Service-5']);

  while( my $hash = shift(@{$values} ) ){
                                            put_log('INFO', msg=>"Result ==>
$hash->{'OBJECT_ID'} : $hash->{'OBJECT_NAME'} : $hash->{'COMMENT'} ");

while( my $hashref = shift(@{$hash->{'MEMBERS'}} ) ){
                                         while( my($key, $val ) = each(
%{$hashref} ) ){
          put_log('INFO',msg=> "$key\t$val" );
                                                }       }      }
```

   GET GROUP SERVICE

```
( $result, $values )  =
$service->getGroupObjects('OBJECT_NAME'=>['Group-1','Group-2','Group-3']);
               while( my $hash = shift(@{$values} ) ){
                                                          put_log('INFO',
msg=>"Result ==> $hash->{'OBJECT_ID'} : $hash->{'OBJECT_NAME'} :
$hash->{'COMMENT'} ");           while( my $tmp = shift(@{$hash->{'MEMBERS'}}
 ) ){
put_log('INFO',msg=> "$tmp" );
                                     }  }
```

   GET GLOBAL SERVICE

```
( $result, $values )  =
$service->getGlobalObjects('OBJECT_NAME'=>['Poly-1','Poly-2','Poly-3','Poly-4']);
           while( my $hash = shift(@{$values} ) ){
                                                    put_log('INFO', msg=>"Result
 ==> $hash->{'OBJECT_ID'} : $hash->{'OBJECT_NAME'} : $hash->{'COMMENT'} ");
           while( my $hashref = shift(@{$hash->{'MEMBERS'}} ) ){
                                                    put_log('INFO',msg=> "Result
```

```
--> ".$hashref->{'DOMAIN'}.":".$hashref->{'DEVICE'}.":".$hashref->{'SERVICE'}
);     }
 }
```

3. Log out.

Replace Group-Global Service Objects

This section shows how to use the Perl Client Library to replace Group-Global service objects.

To replace a service object:

1. Log into the Perl client.
2. Replace the object.

   Enter:

```
$result   = $service->replaceGlobalObjects('OBJECT_NAME'=>'Poly-2',
'GROUP_MEMBERS'=>$members, 'COLOR'=>'red' );

  put_log('WARN', msg=>"Error while adding group objects." ) unless( $result
);
```

3. Log out.

Delete All Group-Global Service Objects

This section shows how to use the Perl Client Library to delete all Group-Global service objects.

To delete all service objects:

1. Log into the Perl client.
2. Delete the object.

   Enter:

```
    my $result  = $service->deleteAllObjects();
                                        put_log('WARN', msg=>"Error while deleting
 all objects." ) unless( $result );
```

3. Log out.

Using the Perl Client Library with Device Objects

This section shows how to use the Perl Client Library to read device objects.

NOTE: The COLOR and COMMENT arguments are optional for these procedures.

Read Device Objects

This section shows how to read device objects.

To read device objects:

1. Log into the Perl client.
2. Initialize the device object. Enter:

```
my $device = UTILS::DEVICE->init( 'SOAP'=>$connect );
```

3. Read the object. Enter:

```
my( $result, $values )  =
$device->getDeviceObjects('OBJECT_NAME'=>['sweepea','droopy'] );

while( my $hash = shift(@{$values} ) ){

print "\n-------------------------\n";

while( my ($key, $val ) = each( %{$hash } ) ){

put_log('INFO',msg=> "KEY : $key\t    VALUE : $val" ); }


}
```

4. Log out.

Troubleshooting Common Perl API issues

This section describes how to address common Perl NSM API issues.

Problem 1:

Issue: Unable to connect to NSM using the script.

Solution: Check the connectivity between the client and the NSM server if NSM GuiSvr port is responding.

```
Successful Connection:
# telnet 10.205.1.251 8443
Trying 10.205.1.251...
Connected to 10.205.1.251.
Escape character is '^]'.
```

Problem 2:

Issue: The connection is fine, but Perl API is not connecting.

Solution: Enable SOAP traces to understand the cause of the problem.

To enable SOAP trace, change the following declaration in the script.

```
FROM:
use SOAP::Lite;
```

```
TO:
use SOAP::Lite +trace => 'all';
```

The above change provides detailed connection data that provides more details on the problem.

Problem 3:

Issue: If connection fails and if we notice the following output in the script output.

```
SOAP::Transport::HTTP::Client::send_receive: HTTP::Request=HASH(0xa69b9f8)
SOAP::Transport::HTTP::Client::send_receive: POST
 https://10.205.1.253:8443/axis2/services/SystemService HTTP/1.1

<SNIP>
<SOAPAction: "http://juniper.net/webproxy/systemservice#LoginRequest"
<SNIP>
<SOAP::Transport::HTTP::Client::send_receive: HTTP::Response=HASH(0xa82c8b8)
SOAP::Transport::HTTP::Client::send_receive: 500 Can't connect to 10.205.1.253:8443
 (certificate verify failed) Content-Type: text/plain
Client-Date: Tue, 03 Dec 2013 07:01:47 GMT
Client-Warning: Internal response

Can't connect to 10.205.1.253:8443 (certificate verify failed)
```

Solution: The issue provided above is due to the failure of HTTPS certificate that the NSM server hosts is not being validated on client where the script is being run.

To fix the issue, add the following statement at the start of the script after the declaration.

```
Add: BEGIN { $ENV{PERL_LWP_SSL_VERIFY_HOSTNAME} = 0 }
Example:
#!/usr/bin/perl

use SOAP::Lite;
#use SOAP::Lite +trace => 'all';
use MAIN::NSM;
use UTILS::ADDRESS;
use Data::Dumper;

BEGIN { $ENV{PERL_LWP_SSL_VERIFY_HOSTNAME} = 0 }
```

Example:

```
Get All Address Script:

#!/usr/bin/perl

use SOAP::Lite;
use MAIN::NSM;
use UTILS::ADDRESS;

BEGIN { $ENV{PERL_LWP_SSL_VERIFY_HOSTNAME} = 0 }

# Login to NSM server
print("Please enter NSM device ip address:");
my $host = '<NSM SERVER IP>';
print "\nPlease enter USERNAME:";
 $USERNAME = '<NSM Username>';
 print "\nPlease enter passwd for $USERNAME (no echo):";
```



```
 $PASSWORD = '<NSM Password>';
print "Logging in to NSM server at $host...\n";
 $connect = MAIN::NSM->new( 'HOST' => "$host", 'PORT' => "$port", );
 my $connect_info =
 $connect->login( 'UNAME' => $USERNAME, 'PASSWD' => $PASSWORD );
unless ($connect_info) { die( print "Could not connect to $host\n" ); }
# Initialize ADDRESS MODULE
$address = UTILS::ADDRESS->init( 'SOAP' => $connect );
unless ($connect_info) { die( print "Could not connect to $host\n" ); }
# Retrieve all Address objects from all domains.
my $t = shift;
my ( $result, $values ) ;
 ( $result, $values )  = $address->getAllObjects('DOMAIN_ID'=>'1' );
put_log('WARN', msg=>"Error while reading Host objects." ) unless( $result );

 while( my $hash = shift(@{$values} ) ){
   put_log('INFO', msg=>"Result ==> $hash->{'OBJECT_ID'} : $hash->{'OBJECT_NAME'}
 : $hash->{'DOMAIN'} : $hash->{'ZONE'}");    }
cleanup();
# SUBROUTINES
sub cleanup {
   # Logout of NSM server
      $connect->logout;
 print "\n";
     exit(0); }
```

PART 4

Using the NSM API from a Java Client

This part explains how to access and use the NSM API from a Java client. The Java client samples in this section use the data binding Java classes generated by Axis2. This sample code is also located in the directory **$NSROOT/GuiVar/lib/webproxy/client** on the NSM server.

- Using APIs for Authentication on page 99
- Using APIs for Policy Management on page 101
- Using APIs for Shared Object Management on page 111
- Using APIs for Job Management on page 117
- Using APIs for Device Management on page 125
- Using APIs for DeviceObj Management on page 127

CHAPTER 9

Using APIs for Authentication

This chapter shows how to use the NSM APIs to log into and log out of the system.

This chapter contains the following sections:

- Login on page 99
- Logout on page 100

Login

This API sample code shows how to log into the NSM server.

```
/**
     * Prerequisite:
     *
     * @throws Exception
     */
    public void setUp() throws Exception {

        if (PolicyAssignmentTest.setUpDone) return;

        Properties properties = null;

        webDir = System.getProperty("WEBDIR");

        if (webDir == null) {
            webDir = System.getProperty("user.dir") + File.separator + "..";
            System.err.println("WEBDIR is not defined, using the default one " +
 webDir);
        }
        File argsCandidate = new File(webDir + File.separator + "client" +
             File.separator + "Properties.txt");

        if (null != argsCandidate) {
            properties = new Properties();
            FileInputStream fin = new FileInputStream(argsCandidate);
            properties.load(fin);
            fin.close();
        }

        String trustStore = (String) properties.get("javax.net.ssl.trustStore");

       String trustStorePath = webDir + File.separator + trustStore.replace('/',

             File.separatorChar);
```

```
        System.setProperty("javax.net.ssl.trustStore", trustStorePath);
        System.setProperty("javax.net.ssl.trustStorePassword", "changeit");
        String log4jProperties = (String) properties.get("log4j.configuration");


        String serverUrl = (String) properties.get("server.url");
        Properties props = new Properties();
       props.load(new FileInputStream(webDir + File.separator + log4jProperties));

        PropertyConfigurator.configure(props);

        PolicyAssignmentTest.stub = new DataCentricServiceStub(serverUrl +
             "/axis2/services/DataCentricService");
        PolicyAssignmentTest.jobServiceStub = new JobServiceStub(serverUrl +
             "/axis2/services/JobService");

        /**
         *  Login to the system
         */
        SystemServiceStub systemServiceStub = new SystemServiceStub(serverUrl +

             "/axis2/services/SystemService");

        LoginRequest loginRequest = new LoginRequest();
        loginRequest.setUserName("super");
        loginRequest.setDomainName("global");
        loginRequest.setPassword("netscreen");
        LoginResponseType loginResponse =
             systemServiceStub.LoginRequest(loginRequest).getLoginResponse();
        if ((LoginStatusCodeType.Success).equals(loginResponse.getLoginStatus().

              getStatus())) {
            PolicyAssignmentTest.authToken = loginResponse.getAuthToken();
        }

        PolicyAssignmentTest.setUpDone = true;

    }
```

Logout

This API sample code shows how to log out from the NSM server.

```
public void testLogout() {
  try {
   LogoutRequest logoutRequest = new LogoutRequest();
       logoutRequest.setAuthToken(authToken);

       System.out.println("\nAuthToken: " + authToken.getToken() + " logging out
 ...");
       stub.LogoutRequest(logoutRequest);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }
```

CHAPTER 10

Using APIs for Policy Management

This chapter shows how to use the NSM APIs to manage policies.

This chapter contains the following sections:

- Create a New Policy on page 101
- Update an Existing Policy on page 103
- Delete a Policy on page 104
- Get a List of Policies on page 105
- Get a Policy on page 106
- Assign a Policy to a Device on page 106
- Remove a Policy Assignment on page 107
- Delete Rule in Policy on page 108

Create a New Policy

This section provides Data Centric Service API sample code that creates a new policy.

testPolicyInsert.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<nsmpolicy>
  <name_>test100</name_>
  <accesstype>regular</accesstype>
  <rulebases>
    <firewall>&1.rb_firewall.?????????rb_test100_130q6f9fs</firewall>
  </rulebases>
</nsmpolicy>
```

testRuleBaseFirewall Insert.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<rb_firewall>
  <name_>rb_test100_130q6f9fs</name_>
  <rules_collection>
    <rules>
      <name_></name_>
      <direction>
        <global>false</global>
        <from_zone>trust</from_zone>
        <to_zone>untrust</to_zone>
      </direction>
```

```
      <dialupvpn>
        <enabled>false</enabled>
        <src-or-dst>none</src-or-dst>
      </dialupvpn>
      <src_addr_collection>
        <src_addr>any</src_addr>
      </src_addr_collection>
      <dst_addr_collection>
        <dst_addr>any</dst_addr>
      </dst_addr_collection>
      <service_collection>
        <service>any</service>
      </service_collection>
      <action>
        <deny>null</deny>
      </action>
      <preferred-id>3</preferred-id>
      <count>
        <disabled>null</disabled>
      </count>
      <target_collection>
        <target>any</target>
      </target_collection>
      <auths>
        <no-auth>null</no-auth>
      </auths>
      <traffic>
        <enabled>false</enabled>
      </traffic>
    </rules>
  </rules_collection>
</rb_firewall>
```

Sample Code

```
 * Inserts DB with nsmpolicy object with a default firewall rule base specified
in XML data.
       *
       * Prerequisite:
       *  No nsmpolicy and rule base are allowed with the same name as that in
the input.*/
      public void testInsertNsmPolicyObject() {
          try {
              File nsmPolicyFile = new File(webDir + File.separator + pathOfInput
 +
                    "/Input/" + "testPolicyInsert.xml");
               File rbFile = new File(webDir + File.separator + pathOfInput +
"/Input/" +
                    "testRuleBaseFirewallInsert.xml");

               System.out.println("Running testInsertNsmPolicyObject()");

               //creates an object of ModifyObjectViewRequest
               ModifyObjectViewRequest request = new ModifyObjectViewRequest();
               request.setAuthToken(DataCentricServiceTest.authToken);

               //creates an object of ModifyViewCommandType
               ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
               InsertObjectViewType insertObject = new InsertObjectViewType();
               modifyCmd.setInsertObject(insertObject);
               insertObject.setCategory("nsmpolicy");
```

```
                insertObject.setDomainId(new UnsignedShort("1"));

                //reads the policy in XML format from file "testPolicyInsert.xml"
                XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
                XMLStreamReader parser = xmlInputFactory.createXMLStreamReader
                     (new FileInputStream(nsmPolicyFile));
                StAXOMBuilder builder = new StAXOMBuilder(parser);
                OMElement ome  = builder.getDocumentElement();
                ObjectDataType objData = new ObjectDataType();
                objData.setData(this.createOpaqueDataType(ome));
                insertObject.setObjecData;

                //creates another object of ModifyViewComamndType
                ModifyViewCommandType insertRulebaseCmd = new
ModifyViewCommandType();
                InsertObjectViewType insertRulebaseObject = new
InsertObjectViewType();
                insertRulebaseCmd.setInsertObject(insertRulebaseObject);
                insertRulebaseObject.setCategory("rb_firewall");
                insertRulebaseObject.setDomainId(new UnsignedShort("1"));

                //reads the rulebase in XML format from file
"testRuleBaseFirewallInsert.xml"
                parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(rbFile));
                builder = new StAXOMBuilder(parser);
                OMElement omeRulebase  = builder.getDocumentElement();
                ObjectDataType rbData = new ObjectDataType();
                rbData.setData(this.createOpaqueDataType(omeRulebase));
                insertRulebaseObject.setObjecData(rbData);

                //inserts the rulebase first, then insert the policy
                request.addCommand(insertRulebaseCmd);
                request.addCommand(modifyCmd);

                //invokes the service
                ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
                print(response);
                assertTrue(response.getStatus() == StatusCodeType.Success);

            }catch (Exception e) {
                e.printStackTrace();
            }
        }
```

Update an Existing Policy

The following Data Centric Service API code sample updates an existing service.

```
 * Updates the service in the rule base with a predefined service object.
     */
    public void testUpdateNodeRequest_Rulebase() {
        try {
            System.out.println("Running testUpdateNodeRequest_Rulebase()");

            //creates an object of ModifyObjectViewRequest
            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
```

```
            //creates an object of ModifyViewCommandType
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();

            //specifies the device from which to the policy is unassigned
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("rb_firewall");
            objectId.setDomainId(new UnsignedShort("1"));
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectName("rb_test100_130q6f9fs");
            objectId.setObjectIdOrName(objIdOrName);

            //creates an object of UpdateObjectViewType
            UpdateObjectViewType updateObject = new UpdateObjectViewType();
            updateObject.setObjectIdentifier(new ObjectIdentifierType[]
{objectId});
            modifyCmd.setUpdateObject(updateObject);
          NodeModificationType nodeModificationType = new NodeModificationType();

            PathValueType pathValueType = new PathValueType();
            nodeModificationType.setUpdateNode(pathValueType);

            //changes the first service in the service collection to the service
 specified

pathValueType.setXpath("./rules_collection/rules[1]/service_collection/service[1]");

            pathValueType.setValue(this.createOpaqueDataType
                 ("<service>&0.service.15</service>"));
            updateObject.setObjectModification(new ObjectModificationType());

updateObject.getObjectModification().addModification(nodeModificationType);

            request.addCommand(modifyCmd);

            //invokes the service
            ModifyObjectViewResponse response =
                 DataCentricServiceTest.stub.ModifyObjectViewRequest(request);
            print(response);
            assertTrue(response.getStatus() == StatusCodeType.Success);

        }catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Delete a Policy

The following Data Centric Service API code sample deletes the NSM policy and associated rules.

```
public void testDeleteNsmpolicyObject() {
        try {
            System.out.println("Running testDeleteNsmpolicyObject()");

            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);

            //creates an object of ModifyViewCommandType
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            DeleteObjectViewType deleteObject = new DeleteObjectViewType();
```

```
            //specifies the rulebase to be deleted
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("rb_firewall");
            objectId.setDomainId(new UnsignedShort("1"));
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectName("rb_test100_130q6f9fs");
            objectId.setObjectIdOrName(objIdOrName);

            //specifies the policy to be deleted
            ObjectIdentifierType nsmpolicyId = new ObjectIdentifierType();
            nsmpolicyId.setCategory("nsmpolicy");
            nsmpolicyId.setDomainId(new UnsignedShort("1"));
            ObjectIdOrNameType nsmpolicyIdOrName = new ObjectIdOrNameType();
            nsmpolicyIdOrName.setObjectName("test100");
            nsmpolicyId.setObjectIdOrName(nsmpolicyIdOrName);
            deleteObject.addObjectIdentifier(nsmpolicyId);
            deleteObject.addObjectIdentifier(objectId);
            modifyCmd.setDeleteObject(deleteObject);
            request.addCommand(modifyCmd);

            //invokes the service
            ModifyObjectViewResponse response =
                 DataCentricServiceTest.stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Get a List of Policies

This Data Centric Service API code sample gets a list of all policies in one domain.

```
    /**
     * Gets all the policy objects in one domain.
     * <p/>
     */
    public void testGetAllPolicyObject() {
        try {
            System.out.println("Running testGetAllPolicyObject()");

            GetObjectViewByCategoryRequest request = new
GetObjectViewByCategoryRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
            request.setDomainId(new UnsignedShort(1));
            request.setCategory("nsmpolicy");

            GetObjectViewByCategoryResponse response =

DataCentricServiceTest.stub.GetObjectViewByCategoryRequest(request);

            DataCentricServiceTest.print(response.getObject());
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Get a Policy

This Data Centric Service API code sample gets a specific policy.

```
    /**
     * Gets a single policy object.
     * <p/>
     * Prerequisite:
     * A policy object has been added in NSM.
     */
    public void testGetPolicyObject() {
        try {
            System.out.println("Running testGetPolicyObject()");

            GetObjectViewByIdRequest request = new GetObjectViewByIdRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
            ObjectIdentifierType oid = new ObjectIdentifierType();
            oid.setDomainId(new UnsignedShort("1"));
            oid.setCategory("nsmpolicy");

            ObjectIdOrNameType choice = new ObjectIdOrNameType();
            choice.setObjectName("test");
            oid.setObjectIdOrName(choice);

            request.addObjectIdentifier(oid);

            GetObjectViewByIdResponse response =
                 DataCentricServiceTest.stub.GetObjectViewByIdRequest(request);

            DataCentricServiceTest.print(response.getObject());
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Assign a Policy to a Device

This Data Centric Service API code sample assigns a policy.

```
/**
     * Prerequisite: there is a device with id 2
     *               there is a policy with name test
     */
    public void testAssignPolicy2Device() {
        try {
            System.out.println("Running testAssignPolicy2Device()");

            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(PolicyAssignmentTest.authToken);

            //creates an object of ModifyViewCommandType:
:
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            DeleteObjectViewType deleteObject = new DeleteObjectViewType();

            //specifies the device to assign the policy:
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("deviceobj");
```

```
            objectId.setDomainId(new UnsignedShort("1"));
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectId(new UnsignedInt(0));
            objectId.setObjectIdOrName(objIdOrName);

            //create an object of UpdateObjectViewType:

            UpdateObjectViewType updateObject = new UpdateObjectViewType();
            updateObject.setObjectIdentifier(new ObjectIdentifierType[]
{objectId});
            modifyCmd.setUpdateObject(updateObject);
          NodeModificationType nodeModificationType = new NodeModificationType();

            PathValueType pathValueType = new PathValueType();
            nodeModificationType.setUpdateNode(pathValueType);

            //adds the policy ID to the deviceobj:

      pathValueType.setXpath("./nsmpolicy-id"):
            pathValueType.setValue(this.createOpaqueDataType
                 ("<nsmpolicy-id>&1.nsmpolicy.?????????test </nsmpolicy-id>"));
            updateObject.setObjectModification(new ObjectModificationType());

updateObject.getObjectModification().addModification(nodeModificationType);

            request.addCommand(modifyCmd);

            //invokes the service:

                 ModifyObjectViewResponse response =
                 PolicyAssignmentTest.stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Remove a Policy Assignment

This Data Centric Service API code sample removes a policy assignment from a device.

```
/**
     * Prerequisite: there is a device with id 2
     *               there is a policy with name test
     */
    public void testUnAssignPolicy2Device() {
        try {
            System.out.println("Running testUnAssignPolicy2Device()");

            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(PolicyUnAssignmentTest.authToken);

            //creates an object of ModifyViewCommandType:

            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();


            //specifies the device from which to unassign the policy:
```

```
                ObjectIdentifierType objectId = new ObjectIdentifierType();
                objectId.setCategory("deviceobj");
                objectId.setDomainId(new UnsignedShort("1"));
                ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
                objIdOrName.setObjectId(new UnsignedInt(0));
                objectId.setObjectIdOrName(objIdOrName);

                //creates an object of UpdateObjectViewType:

                UpdateObjectViewType updateObject = new UpdateObjectViewType();
                updateObject.addObjectIdentifier(objectId);
                modifyCmd.setUpdateObject(updateObject);
              NodeModificationType nodeModificationType = new NodeModificationType();

                nodeModificationType.setDeleteNode("./nsmpolicy-id");

                updateObject.setObjectModification(new ObjectModificationType());

updateObject.getObjectModification().addModification(nodeModificationType);

                request.addCommand(modifyCmd);

                //invokes the service:

                ModifyObjectViewResponse response =
                     PolicyUnAssignmentTest.stub.ModifyObjectViewRequest(request);
                assertTrue(response.getStatus() == StatusCodeType.Success);

            } catch (Exception e) {
                e.printStackTrace();
            }
        }
```

Delete Rule in Policy

This Data Centric Service API sample code removes Rule from the Rule base (example: rb_firewall). This sample code needs a Rule ID as input for deleting the Rule from the Rulebase.

Sample Code

```
public void testDeleteRule() {
        try {

         System.out.println("Running testDeleteRule");
         //create an object of ModifyObjectViewRequest
         ModifyObjectViewRequest request = new ModifyObjectViewRequest();
         request.setAuthToken(DeleteRuleRequestTest.authToken);
         ObjectIdentifierType objectId = new ObjectIdentifierType();
         objectId.setCategory("rb_firewall");
         objectId.setDomainId(new UnsignedShort("1"));
         ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
         objIdOrName.setObjectName("rb_test_185pqmgo6");
         objectId.setObjectIdOrName(objIdOrName);
         //create an object of ModifyViewCommandType
         ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
         //Create an object for DeleteRuleViewType
         DeleteRuleViewType deleteRuleObject = new DeleteRuleViewType();
         deleteRuleObject.setObjectIdentifier(objectId);
         deleteRuleObject.setObjecData(new ObjectDataType());
```

```
         deleteRuleObject.setPreferredId("2");
         modifyCmd.setDeleteRuleObject(deleteRuleObject);
         request.addCommand(modifyCmd);
         //invoke the service
         ModifyObjectViewResponse response =
DeleteRuleRequestTest.stub.ModifyObjectViewRequest(request);
          assertTrue(response.getStatus() == StatusCodeType.Success);
        }catch (Exception e) {
 e.printStackTrace();
        }
    }
```

CHAPTER 11

Using APIs for Shared Object Management

The following sections show how to use the NSM APIs to manage shared objects.

- Insert a Shared Object on page 111
- Replace a Shared Object on page 112
- Delete a Shared Object on page 114
- Get a List of Shared Objects on page 114
- Get a Shared Object on page 115
- Associate Shared Object to its Group Object on page 116

Insert a Shared Object

This sample shows how to insert a shared object.

The following XML documentation is the input for the Data Centric Service API sample code.

testAddressInsert.xml

```
<?xml version="1.0" encoding="UTF-8"?>

    <address>
   <name_>AddrA</name_>
   <address>
        <zone>trust</zone>
        <address>
          <subnet>
            <ip>1.1.1.8</ip>
            <netmask>21</netmask>
          </subnet>
        </address>
      </address>
      <comment>AddrA object</comment>
 </address>
```

Sample Code

```
  * Inserts an Address object
     */
    public void testInsertAddressObject() {
```

```
        try {

            File addressFile = new File(webDir + File.separator + pathOfInput +

                 "/Input/" + "testAddressInsert.xml");

            System.out.println("Running testInsertAddressObject()");

            //creates an object of ModifyObjectViewRequest:

            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            DomainIdOrNameType domain = new DomainIdOrNameType();
            domain.setDomainId(new UnsignedShort("1"));

            InsertObjectViewType insertObject = new InsertObjectViewType();
            insertObject.setCategory("address");
            insertObject.setDomainId(new UnsignedShort("1"));

            //reads the address object in XML format from the file
"testAddressInsert.xml":

            XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
            XMLStreamReader parser =
                 xmlInputFactory.createXMLStreamReader(new
FileInputStream(addressFile));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();

            insertObject.setObjecData(new ObjectDataType());
            insertObject.getObjecData().setData(this.createOpaqueDataType(ome));

            modifyCmd.setInsertObject(insertObject);
            request.addCommand(modifyCmd);

            //invokes the service:

            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Replace a Shared Object

The following Data Centric Service API sample code replaces a shared address object.

The following XML documentation is the input for the Data Centric Service API sample code shown below.

testAddressReplace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
   <address>
   <name_>AddrA</name_>
   <address>
```

```
        <zone>trust</zone>
        <address>
          <subnet>
            <ip>1.1.1.7</ip>
            <netmask>21</netmask>
          </subnet>
        </address>
      </address>
 </address>
```

Sample Code

```
 * Tests replacing an Address object
    */
   public void testReplaceAddressObject() {
       try {

           File addressFile = new File(webDir + File.separator + pathOfInput +

                "/Input/" + "testAddressReplace.xml");
           System.out.println("Running testReplaceAddressObject()");


           ModifyObjectViewRequest request = new ModifyObjectViewRequest();
           request.setAuthToken(DataCentricServiceTest.authToken);

           ModifyViewCommandType modifyCmd = new ModifyViewCommandType();

           //specifies the address object to replace:

           ObjectIdentifierType objectId = new ObjectIdentifierType();
           objectId.setCategory("address");
           objectId.setDomainId(new UnsignedShort("1"));
           ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
           //objIdOrName.setObjectId(new UnsignedInt("1"));
           objIdOrName.setObjectName("AddrA");
           objectId.setObjectIdOrName(objIdOrName);

           ReplaceObjectViewType replaceObject = new ReplaceObjectViewType();

           //reads the new address in XML format from the file
"testAddressReplace.xml":

           XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
           XMLStreamReader parser =
                xmlInputFactory.createXMLStreamReader(new
FileInputStream(addressFile));
           StAXOMBuilder builder = new StAXOMBuilder(parser);
           OMElement ome = builder.getDocumentElement();

           replaceObject.setObjecData(new ObjectDataType());
          replaceObject.getObjecData().setData(this.createOpaqueDataType(ome));


           replaceObject.setObjectIdentifier(objectId);
           modifyCmd.setReplaceObject(replaceObject);
           request.addCommand(modifyCmd);

           ModifyObjectViewResponse response =
                DataCentricServiceTest.stub.ModifyObjectViewRequest(request);
           assertTrue(response.getStatus() == StatusCodeType.Success);
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Delete a Shared Object

The following Data Centric Service API sample code deletes a shared address object.

```
     * Deletes an Address object
     */
    public void testDeleteAddressObject() {
        try {
            System.out.println("Running testDeleteAddressObject()");

            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);

            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            DeleteObjectViewType deleteObject = new DeleteObjectViewType();
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("address");
            objectId.setDomainId(new UnsignedShort("1"));
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectName("AddressObjectA");
            objectId.setObjectIdOrName(objIdOrName);

            deleteObject.addObjectIdentifier(objectId);
            modifyCmd.setDeleteObject(deleteObject);
            request.addCommand(modifyCmd);

            ModifyObjectViewResponse response =
                 DataCentricServiceTest.stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Get a List of Shared Objects

The following Data Centric Service API sample code gets a list of shared objects.

```
  * Gets an object by category.     */
    public void testGetCategoryObject() {
        try {
            System.out.println("Running testGetCategoryObject()");

            GetObjectViewByCategoryRequest request = new
GetObjectViewByCategoryRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
            request.setCategory("address");
            request.setDomainId(new UnsignedShort("1"));

            GetObjectViewByCategoryResponse response =

DataCentricServiceTest.stub.GetObjectViewByCategoryRequest(request);
```

```
            System.out.println("Status=" + response.getStatus());
            DataCentricServiceTest.print(response.getObject());

            //gets the first address object returned:

            InputStream inputStream =

response.getObject()[0].getObjectData().getData().getBase64Binary().getInputStream();


            //uses XPath to get the IP address of the address object without
deserializing the whole object:

            XPathFactory factory = XPathFactory.newInstance();
            XPath xpath = factory.newXPath();
            XPathExpression expr =
xpath.compile("/address/address/address/subnet/ip/text()");
            String ipAddr = expr.evaluate(new InputSource(inputStream));
            System.out.println("The ip address is " + ipAddr);
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Get a Shared Object

The following Data Centric Service API sample code gets a shared address object.

```
* Gets a single Address Object
     * <p/>
     * Prerequisite:
     * An address object has been added in NSM
     */
    public void testGetAddressObject() {
        try {
            System.out.println("Running testGetAddressObject()");

            GetObjectViewByIdRequest request = new GetObjectViewByIdRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
            ObjectIdentifierType oid = new ObjectIdentifierType();
            oid.setDomainId(new UnsignedShort("1"));
            oid.setCategory("address");

            ObjectIdOrNameType choice = new ObjectIdOrNameType();
            choice.setObjectName("AddrA");
            oid.setObjectIdOrName(choice);

            request.addObjectIdentifier(oid);

            GetObjectViewByIdResponse response =
                 DataCentricServiceTest.stub.GetObjectViewByIdRequest(request);

            DataCentricServiceTest.print(response.getObject());
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Associate Shared Object to its Group Object

The following Data Centric Service API sample code adds a shared object to its group's object.

Sample Code

```
public void testAssociateSharedObject() {
        try {
            System.out.println("Running testAssociateSharedObject ()");
            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();

            //specify the address object to update
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("address ");
            objectId.setDomainId(new UnsignedShort("1"));
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectName("AddressGroup");
            objectId.setObjectIdOrName(objIdOrName);

            UpdateObjectViewType updateObject = new UpdateObjectViewType();
            updateObject.setObjectIdentifier(new ObjectIdentifierType[]
{objectId});
            modifyCmd.setUpdateObject(updateObject);

          NodeModificationType nodeModificationType = new NodeModificationType();

            PathValueType pathValueType = new PathValueType();
            nodeModificationType.setUpdateNode(pathValueType);

            updateObject.setObjectModification(new ObjectModificationType());

updateObject.getObjectModification().addModification(nodeModificationType);

            //update the ip address at the node specified by the path
            pathValueType.setXpath(".//group/members_collection/members[1]");
            pathValueType.setValue(this.createOpaqueDataType
(“<members>&amp;1.address.403</members>”));
            request.addCommand(modifyCmd);
            ModifyObjectViewResponse response =
DataCentricServiceTest.stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
```

CHAPTER 12

Using APIs for Job Management

These examples show how the Job Service API is used to manage jobs.

- Get a Job Result on page 117
- Import a List of Devices on page 118
- Update a List of Devices on page 119
- Get a Configuration Summary on page 119
- Get a Running Configuration on page 120
- Get the Delta Configuration on page 121
- Cancel a Job Request on page 121
- Validate Device Configuration on page 122

Get a Job Result

This Job Service API code sample gets status information about current jobs.

```
public void getJobResult(String jobName, JobArgsType jobArgs) {
     try {
      StatusCodeType opStatus = StatusCodeType.Failure;
      GetJobStatusRequest jobStatusRequest = new GetJobStatusRequest();
   jobStatusRequest.setAuthToken(authToken);
   jobStatusRequest.setDomainId(jobArgs.getDomainId());
   jobStatusRequest.setJobName(jobName);

   JobStatusType jobStatus = null;
   while (true) {
    GetJobStatusResponse jobStatusResponse =
stub.GetJobStatusRequest(jobStatusRequest);
    opStatus = jobStatusResponse.getStatus();
    JobResponseType jobResponseList[] = jobStatusResponse.getJobStatus();
    for (int i = 0; i< jobResponseList.length; i++) {
     if (jobName.equals(jobResponseList[i].getJobName())) {
      jobStatus = jobResponseList[i].getStatus();
      System.out.println("JobName: " + jobName + " opStatus: " +
                  opStatus + " jobStatus: " + jobStatus.toString());
     }
    }
    assertTrue(opStatus == StatusCodeType.Success);

    if ((jobStatus == JobStatusType.COMPLETEDWITHSUCCESS) ||
              (jobStatus == JobStatusType.COMPLETEDWITHFAILURE)) {
```

```
    break;
   } else {
    Thread.sleep(5000);
   }
  }
  assertTrue(jobStatus == JobStatusType.COMPLETEDWITHSUCCESS);

  GetJobResultRequest jobResultRequest = new GetJobResultRequest();
  jobResultRequest.setAuthToken(authToken);
  jobResultRequest.setDomainId(jobArgs.getDomainId());
  jobResultRequest.setJobName(jobName);
  GetJobResultResponse jobResultResponse =
           stub.GetJobResultRequest(jobResultRequest);
  opStatus = jobResultResponse.getStatus();
  JobResponseType jobResultList[] = jobResultResponse.getJobResponse();
  for (int i = 0; i < jobResultList.length; i++) {
   if (jobName.equals(jobResultList[i].getJobName())) {
    jobStatus = jobResultList[i].getStatus();
    String jobResult = jobResultList[i].getExplanation().toString();
    System.out.println("JobName: " + jobName + " jobResult: " + jobResult);
   }
  }
  assertTrue(opStatus == StatusCodeType.Success);
    } catch (Exception e) {
     e.printStackTrace();
    }
   }
```

Import a List of Devices

This Job Service API code example imports a list of devices.

```
public void testImportDeviceRequest() {
     try {
      File importDeviceInput = new File(webDir + File.separator + pathOfInput +

            "/Input/" + "testImportDevice.txt");
      BufferedReader reader =
            new BufferedReader(new InputStreamReader(new
FileInputStream(importDeviceInput)));

   ImportDeviceRequest importDeviceRequest = new ImportDeviceRequest();
   String jobName = getUniqueJobId(import_device);

   System.out.println("importDeviceRequest: jobName " + jobName);
   JobRequestType jobRequest = getJobRequest(jobName, reader);
   importDeviceRequest.setJobRequest(jobRequest);
   importDeviceRequest.setAuthToken(authToken);

   ImportDeviceResponse importResponse =
            stub.ImportDeviceRequest(importDeviceRequest);
   JobResponseType jobResponse = importResponse.getJobResponse();
   StatusCodeType opStatus = importResponse.getStatus();
   String jobID = jobResponse.getJobName();
   String status = jobResponse.getStatus().toString();
   System.out.println("JobName: " + jobID + " opStatus: " + opStatus +
            " status: " + status);
   assertTrue(opStatus == StatusCodeType.Success);

   getJobResult(jobName, jobRequest.getJobArgs());
```

```
    } catch (Exception e) {
     e.printStackTrace();
    }
 }
```

Update a List of Devices

This Job Service API code sample updates a device list. The devices were imported earlier.

```
public void testUpdateDeviceRequest() throws IOException {
     try {
      File updateDeviceInput = new File(webDir + File.separator + pathOfInput +

            "/Input/" + "testUpdateDevice.txt");
      BufferedReader reader =
            new BufferedReader(new InputStreamReader(new
FileInputStream(updateDeviceInput)));

   UpdateDeviceRequest updateDeviceRequest = new UpdateDeviceRequest();
   String jobName = getUniqueJobId(update_device);

   System.out.println("updateDeviceRequest: jobName " + jobName);

   JobRequestType jobRequest = getJobRequest(jobName, reader);
   updateDeviceRequest.setJobRequest(jobRequest);
   updateDeviceRequest.setAuthToken(authToken);

   UpdateDeviceResponse updateResponse =
            stub.UpdateDeviceRequest(updateDeviceRequest);
   JobResponseType jobResponse = updateResponse.getJobResponse();
   StatusCodeType opStatus = updateResponse.getStatus();
   String jobID = jobResponse.getJobName();
   String status = jobResponse.getStatus().toString();
   System.out.println("JobName: " + jobID + " opStatus: " + opStatus +
            " status: " + status);
   assertTrue(opStatus == StatusCodeType.Success);

   getJobResult(jobName, jobRequest.getJobArgs());
     } catch (Exception e) {
      e.printStackTrace();
     }
 }
```

Get a Configuration Summary

This Job Service API code sample gets a configuration summary.

```
public void testGetConfigSummaryRequest() {
     try {
      File configSummaryInput = new File(webDir + File.separator + pathOfInput +

            "/Input/" + "testConfigSummary.txt");
      BufferedReader reader =
            new BufferedReader(new InputStreamReader(new
FileInputStream(configSummaryInput)));

      GetConfigSummaryRequest getConfigSummaryRequest = new
GetConfigSummaryRequest();
```

```
      String jobName = getUniqueJobId(config_summary);

      System.out.println("getConfigSummaryRequest: jobName " + jobName);

      JobRequestType jobRequest = getJobRequest(jobName, reader);
   getConfigSummaryRequest.setJobRequest(jobRequest);
   getConfigSummaryRequest.setAuthToken(authToken);

   GetConfigSummaryResponse configSummaryResponse =
            stub.GetConfigSummaryRequest(getConfigSummaryRequest);
   JobResponseType jobResponse = configSummaryResponse.getJobResponse();
   StatusCodeType opStatus = configSummaryResponse.getStatus();

   String jobID = jobResponse.getJobName();
   String status = jobResponse.getStatus().toString();
   System.out.println("JobName: " + jobID + " opStatus: " + opStatus +
            " status: " + status);
   assertTrue(opStatus == StatusCodeType.Success);

   getJobResult(jobName, jobRequest.getJobArgs());
     } catch (Exception e) {
      e.printStackTrace();
     }
    }
```

Get a Running Configuration

This Job Service API code sample gets the currently running configuration of previously imported devices.

```
public void testGetRunningConfigRequest() {
     try {
      File runningConfigInput = new File(webDir + File.separator + pathOfInput +

            "/Input/" + "testRunningConfig.txt");
      BufferedReader reader =
            new BufferedReader(new InputStreamReader(new
FileInputStream(runningConfigInput)));

   GetRunningConfigRequest getRunningConfigRequest = new GetRunningConfigRequest();

   String jobName = getUniqueJobId(running_config);

   System.out.println("getRunningConfigRequest: jobName " + jobName);

   JobRequestType jobRequest = getJobRequest(jobName, reader);
   getRunningConfigRequest.setJobRequest(jobRequest);
   getRunningConfigRequest.setAuthToken(authToken);

   GetRunningConfigResponse runningConfigResponse =
            stub.GetRunningConfigRequest(getRunningConfigRequest);
   JobResponseType jobResponse = runningConfigResponse.getJobResponse();
   StatusCodeType opStatus = runningConfigResponse.getStatus();

   String jobID = jobResponse.getJobName();
   String status = jobResponse.getStatus().toString();
   System.out.println("JobName: " + jobID + " opStatus: " + opStatus +
            " status: " + status);
   assertTrue(opStatus == StatusCodeType.Success);
```

```
   getJobResult(jobName, jobRequest.getJobArgs());
     } catch (Exception e) {
      e.printStackTrace();
     }
 }
```

Get the Delta Configuration

This Job Service API code sample gets the delta configuration (difference) between the current configuration and the configuration of previously imported devices.

```
public void testGetDeltaConfigRequest() {
     try {
      File deltaConfigInput = new File(webDir + File.separator + pathOfInput +
            "/Input/" + "testDeltaConfig.txt");
      BufferedReader reader =
            new BufferedReader(new InputStreamReader(new
FileInputStream(deltaConfigInput)));

      GetDeltaConfigRequest getDeltaConfigRequest = new GetDeltaConfigRequest();

      String jobName = getUniqueJobId(delta_config);

      System.out.println("getConfigSummaryRequest: jobName " + jobName);

      JobRequestType jobRequest = getJobRequest(jobName, reader);
   getDeltaConfigRequest.setJobRequest(jobRequest);
   getDeltaConfigRequest.setAuthToken(authToken);

   GetDeltaConfigResponse deltaConfigResponse =
            stub.GetDeltaConfigRequest(getDeltaConfigRequest);
   JobResponseType jobResponse = deltaConfigResponse.getJobResponse();
   StatusCodeType opStatus = deltaConfigResponse.getStatus();

   String jobID = jobResponse.getJobName();
   String status = jobResponse.getStatus().toString();
   System.out.println("JobName: " + jobID + " opStatus: " + opStatus +
            " status: " + status);
   assertTrue(opStatus == StatusCodeType.Success);

   getJobResult(jobName, jobRequest.getJobArgs());
     } catch (Exception e) {
      e.printStackTrace();
     }
 }
```

Cancel a Job Request

This Job Service API code sample shows how to cancel a job request.

```
public void testCancelJobRequest() {
     try {
      File importDeviceInput = new File(webDir + File.separator + pathOfInput +

            "/Input/" + "testImportDevice.txt");
      BufferedReader reader =
            new BufferedReader(new InputStreamReader(new
FileInputStream(importDeviceInput)));
```

```
      //Starts to import
:
   ImportDeviceRequest importDeviceRequest = new ImportDeviceRequest();
   String jobName = getUniqueJobId(import_device);

   System.out.println("importDeviceRequest: jobName " + jobName);
   JobRequestType jobRequest = getJobRequest(jobName, reader);
   importDeviceRequest.setJobRequest(jobRequest);
   importDeviceRequest.setAuthToken(authToken);

   ImportDeviceResponse importResponse =
            stub.ImportDeviceRequest(importDeviceRequest);
   JobResponseType jobResponse = importResponse.getJobResponse();
   StatusCodeType opStatus = importResponse.getStatus();
   String jobID = jobResponse.getJobName();
   String status = jobResponse.getStatus().toString();
   System.out.println("JobName: " + jobID + " opStatus: " + opStatus +
            " status: " + status);
   assertTrue(opStatus == StatusCodeType.Success);

            //Cancels the job:

            System.out.println("Cancel job " + jobID + "now");
            CancelJobRequest cancelJobReq = new CancelJobRequest();
            cancelJobReq.setAuthToken(authToken);
            cancelJobReq.setDomainId(jobRequest.getJobArgs().getDomainId());
            cancelJobReq.setJobName(jobID);
          CancelJobResponse cancelJobResp = stub.CancelJobRequest(cancelJobReq);

            System.out.println("JobName: " + jobID + " opStatus: " +
cancelJobResp.getStatus());


     } catch (Exception e) {
      e.printStackTrace();
     }
 }
```

Validate Device Configuration

This Job Service API code sample shows how to validate device configuration. This sample will display the output of both failure and successful results of Directive.

Sample Code

```
public void testValidateDeviceRequest() {
     try {
File validateDeviceInput = new File(webDir + File.separator + pathOfInput +
"/Input/" + "testValidateDevice.txt");
                 BufferedReader reader =
new BufferedReader(new InputStreamReader(new
FileInputStream(validateDeviceInput)));
     ValidateDeviceRequest validateDeviceRequest = new ValidateDeviceRequest();
 String jobName = getUniqueJobId(“configVerification”);
 System.out.println("validateDeviceRequest: jobName " + jobName);
 JobRequestType jobRequest = getJobRequest(jobName, reader);
 validateDeviceRequest.setJobRequest(jobRequest);
 validateDeviceRequest.setAuthToken(authToken);
ValidateDeviceResponse validateResponse =
```

```
stub.ValidateDeviceRequest(validateDeviceRequest);
 JobResponseType jobResponse = validateResponse.getJobResponse();
 StatusCodeType opStatus = validateResponse.getStatus();
 String jobID = jobResponse.getJobName();
 String status = jobResponse.getStatus().toString();
 System.out.println("JobName: " + jobID + " opStatus: " + opStatus + " status: "
 + status);
 assertTrue(opStatus == StatusCodeType.Success);
 getJobResult(jobName, jobRequest.getJobArgs());
     } catch (Exception e) {
      e.printStackTrace();
     }
 }
```

CHAPTER 13

Using APIs for Device Management

- Retrieve Domains on page 125
- Retrieve the Device List in One Domain on page 125

Retrieve Domains

This System Service API code sample retrieves information about all domains.

```
    public void testGetSystemInfo() {
  try {
   GetSystemInfoRequest getSystemInfoRequest = new GetSystemInfoRequest();
            getSystemInfoRequest.setAuthToken(authToken);


            stub.GetSystemInfoRequest(getSystemInfoRequest);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }
```

Retrieve the Device List in One Domain

This Data Centric Service API code sample retrieves a list of devices in one domain.

```
    /**
     * Gets the IP addresses and the interfaces of all devices in one domain.
      */
    public void testGetDeviceObjectByCategory_Filter() {
        try {
           System.out.println("Running testGetDeviceObjectByCategory_Filter()");


            //creates an object of GetObjectViewByCategoryRequest:
:
            GetObjectViewByCategoryRequest request = new
GetObjectViewByCategoryRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
            request.setCategory("deviceobj");
            request.setDomainId(new UnsignedShort("1"));

          //specifies the filter to retrieve the ip addresse and the interfaces:
```

```
            ViewFilterType filter = new ViewFilterType();
            String subtreeFilter = " <deviceobj
xmlns=\"http://juniper.net/nsm/ADMSchema\">"
                + "  <header>"
                + "    <ip/>"
                + "  </header>"
                + "  <interface_collection/>"
                + " </deviceobj>";

            filter.setFilter(this.createOpaqueDataType(subtreeFilter));
            request.setObjectFilter(filter);

            //invokes the service:

            GetObjectViewByCategoryResponse response =

DataCentricServiceTest.stub.GetObjectViewByCategoryRequest(request);
            System.out.println("Status=" + response.getStatus());
            DataCentricServiceTest.print(response);
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

CHAPTER 14

Using APIs for DeviceObj Management

- Retrieve Device Config with Default Values on page 127
- Modify Device Configuration on page 128
- Manage VPN configuration in DeviceObj Container on page 130
- Manage VPN Configuration in a junos-es Container for Junos OS Devices on page 143
- Manage NAT Configuration in DeviceObj Container on page 147
- Admin User Management on page 152
- Modification of Management interface for Junos OS Devices on page 155
- Assign SNMP Name and SNMP Contact for Junos OS Devices on page 158

Retrieve Device Config with Default Values

The following Data Centric Service API sample code retrieves the Device config with its default values:

Sample Code

```
public void testGetDeviceobjObject() {
try {

System.out.println("Running testGetdeviceObject()");
GetObjectViewByIdRequest request = new GetObjectViewByIdRequest();
            request.setAuthToken(DataCentricServiceTest.authToken);
            ObjectIdentifierType oid = new ObjectIdentifierType();
            oid.setDomainId(new UnsignedShort("1"));

oid.setCategory("deviceobj");

            ObjectIdOrNameType choice = new ObjectIdOrNameType();
            choice.setObjectName("device_name");
            oid.setObjectIdOrName(choice);
            //Retrieve Defaults
            request.setRtveDefaults(true);
            request.addObjectIdentifier(oid);

            GetObjectViewByIdResponse response =
DataCentricServiceTest.stub.GetObjectViewByIdRequest(request);

            DataCentricServiceTest.print(response.getObject());
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }
    }
```

Modify Device Configuration

The following Data Centric Service API sample code replaces device configuration in deviceObj. The following XML documentation is the input for the Data Centric Service API sample code.

testDeviceConfigReplace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<deviceobj>
  <name_>79.78</name_>
  <deployed>true</deployed>
  <type>device</type>
  <nsmpolicy-id>&1.nsmpolicy.66</nsmpolicy-id>
    <header>
    <ip>10.204.79.77</ip>
    <os-version>6.3</os-version>
    <running-os-version>6.3.0r13.0</running-os-version>
    <platform>nsISG1000</platform>
    <productname>NetScreen-ISG1000</productname>
    <ipv6-enabled>true</ipv6-enabled>
    <admin>
      <user-name>netscreen</user-name>
      <password>Rw8sKbpBist2</password>
    </admin>
  </header>
  <admin>
    <user_collection>
      <user>
        <name_>netscreen1</name_>
        <password>nKVUM2rwMUzPcrkG5sWIHdCtqkAibn</password>
        <privilege>root</privilege>
       </user>
    </user_collection>
    <auth>
      <timeout>10</timeout>
    </auth>
    <root-access-console>false</root-access-console>
    <no-internal-command>Please_made_changes_here</no-internal-command>
    <http-re-direct>false</http-re-direct>
  </admin>
 <interface_collection>
    <interface>
      <name_>ethernet1/3.2</name_>
      <ifType>sub</ifType>
      <zone>trust</zone>
      <ip>
        <numbered>
          <ip-address>10.1.2.1</ip-address>
          <netmask>24</netmask>
        </numbered>
      </ip>
      <manage-ip>10.1.2.1</manage-ip>
      <route_deny>false</route_deny>
      <tag-encap>tag</tag-encap>
      <tag>32</tag>
    </interface>
```

```
   </interface_collection>
  <startup>
    <devicefamily>screenos</devicefamily>
    <displaydevicefamily>screenos</displaydevicefamily>
    <connectiontype>static</connectiontype>
    <interfaceipaddress>10.204.79.78</interfaceipaddress>
    <interfacemask>16</interfacemask>
    <interfacegateway>0.0.0.0</interfacegateway>
    <deviceserverip>10.204.79.17</deviceserverip>
    <deviceserverport>7800</deviceserverport>
    <deviceserverip_backup>0.0.0.0</deviceserverip_backup>
    <configletpasswordkeybits>256</configletpasswordkeybits>
    <installerprompt>false</installerprompt>
    <firstconnotp>732239050230592</firstconnotp>
    <connectionmechanism_full>telnet</connectionmechanism_full>
    <usePrimaryForCert>true</usePrimaryForCert>
  </startup>
</deviceobj>
```

Sample Code

```
public void testReplaceDevicConfig() {
    try {
        File deviceFile = new File(webDir + File.separator + pathOfInput +
"/Input/" + "testDeviceConfigReplace.xml");
        System.out.println("Running testReplaceDevicConfig");
        ModifyObjectViewRequest request = new ModifyObjectViewRequest();
        request.setAuthToken(DataCentricServiceTest.authToken);

        ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
        ObjectIdentifierType objectId = new ObjectIdentifierType();
        objectId.setCategory("deviceobj");
        objectId.setDomainId(new UnsignedShort("1"));
        ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
        //objIdOrName.setObjectId(new UnsignedInt("1"));
        objIdOrName.setObjectName("modelscos");
        objectId.setObjectIdOrName(objIdOrName);

        ReplaceObjectViewType replaceObject = new ReplaceObjectViewType();

        XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
        XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(deviceFile));
        StAXOMBuilder builder = new StAXOMBuilder(parser);
        OMElement ome = builder.getDocumentElement();

        replaceObject.setObjecData(new ObjectDataType());
        replaceObject.getObjecData().setData(this.createOpaqueDataType(ome));

        replaceObject.setObjectIdentifier(objectId);
        modifyCmd.setReplaceObject(replaceObject);
        request.addCommand(modifyCmd);

        ModifyObjectViewResponse response =
DataCentricServiceTest.stub.ModifyObjectViewRequest(request);
        assertTrue(response.getStatus() == StatusCodeType.Success);
     }
     catch(Exception e)
     {
       e.printStackTrace();
```

```
    }
  }
```

Limitations

- Cluster Edit is not recommended. The global values that have to be pushed into cluster and the local values that have to be pushed to the member have to be handled from the API client code itself; otherwise it may lead to problems for specific nodes on update.
- If there are values obtained from templates, that is, if a template is assigned to the device, then the values will be overwritten on the full update.
- Full update with default values are not recommended as the default values will be stored in device obj and this will cause issues while editing the Device in GUI.

Manage VPN configuration in DeviceObj Container

- Insert Interface Configuration on page 130
- Replace Interface Configuration on page 131
- Delete Interface Configuration on page 133
- Insert Gateway Configuration on page 133
- Replace Gateway Configuration on page 135
- Delete Gateway Configuration on page 136
- Insert VPN Configuration on page 137
- Replace VPN Configuration on page 138
- Delete VPN Configuration on page 139
- Insert Route Configuration on page 140
- Replace Route Configuration on page 141
- Delete Route Configuration on page 142

This section describes the Data Centric Service API sample codes that can be used to manage VPN configuration in deviceObj.

Insert Interface Configuration

The following sample code inserts interface configuration to the specified device. The sample code needs **testInterfaceInsert.xml** as input.

testInterfaceInsert.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<interface>
      <name_>tunnel.8</name_>
      <mode>nat</mode>
      <mac-address>N/A</mac-address>
      <ifType>tunnel</ifType>
      <zone>trust</zone>
      <ip>
        <numbered>
          <ip-address>10.1.1.4</ip-address>
          <netmask>24</netmask>
```

```
        </numbered>
      </ip>
       <manage-ip>10.1.1.4</manage-ip>
      <route_deny>false</route_deny>
      <acvpn_dynamic_route>false</acvpn_dynamic_route>
</interface>
```

Sample Code

```
public void testInsertInterfaceObject() {
 try{
                   File file = new File(webDir + File.separator + pathOfInput +
"/Input/" + "testInterfaceInsert.xml");
                    System.out.println("Running testInsertInterfaceObject()");
                   //Creating the ModifyObjectViewRequest
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
   request.setAuthToken(authToken);
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
              ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
             objIdOrName.setObjectId(new UnsignedInt("8"));
objIdOrName.setObjectName("SSG");
             objectId.setObjectIdOrName(objIdOrName);
 objectId.setDomainId(new UnsignedInt("1"));
 //Creating a UpdateObjectViewType, this is required since the VPN is inserted
into DeviceObj Section.
AppendDeviceSubNodeViewType appendObject = new AppendDeviceSubNodeViewType();
appendObject.setObjectIdentifier(objectId);
                              appendObject.setNodetype("interface");
                           modifyCmd.setAppendDeviceSubNodeObject(appendObject);

                              XMLInputFactory xmlInputFactory =
XMLInputFactory.newInstance();
            XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();
            ObjectDataType insertObjectData = new ObjectDataType();
            insertObjectData.setData(createOpaqueDataType(ome));
            appendObject.setObjecData(insertObjectData);
            request.addCommand(modifyCmd);
            //invoke the service
            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);
} catch (Exception e) {
            e.printStackTrace();
       }
               }
```

Replace Interface Configuration

The following sample code replaces the existing interface configuration for a specified device. The sample code needs **testInterfaceReplace.xml** as input.

testInterfaceInsert.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<interface>
      <name_>tunnel.2</name_>
```

```
      <mode>nat</mode>
      <mac-address>N/A</mac-address>
      <ifType>tunnel</ifType>
      <zone>trust</zone>
      <ip>
        <numbered>
          <ip-address>20.0.0.4</ip-address>
          <netmask>24</netmask>
        </numbered>
      </ip>
      <manage-ip>20.0.0.4</manage-ip>
      <route_deny>false</route_deny>
      <acvpn_dynamic_route>false</acvpn_dynamic_route>
                                          </interface>
```

Sample Code

```
          public void testReplaceInterfaceObject() {
 try{
          File file = new File(webDir + File.separator + pathOfInput + "/Input/"
 + "testInterfaceReplace.xml");
           System.out.println("Running testReplaceInterfaceObject()");
            //Creating the ModifyObjectViewRequest
            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(authToken);
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("deviceobj");
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectId(new UnsignedInt("0"));
            objIdOrName.setObjectName("SSG");
            objectId.setObjectIdOrName(objIdOrName);
            objectId.setDomainId(new UnsignedInt("1"));
            ReplaceDeviceSubNodeViewType replaceObject = new
ReplaceDeviceSubNodeViewType();
            replaceObject.setObjectIdentifier(objectId);
            replaceObject.setNodetype("interface");
            // The interface in NSM that has to be replaced with the XML input
from the File testInterfaceReplace.xml
            replaceObject.setNodeName(“interface_name”));
            //read the interface object in XML format from the file
"testInterfaceReplace.xml"
            XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
            XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();
          ObjectDataType replaceObjectData = new ObjectDataType();
            replaceObjectData.setData(createOpaqueDataType(ome));
            replaceObject.setObjecData(replaceObjectData);
            modifyCmd.setReplaceDeviceSubNodeObject(replaceObject);
            request.addCommand(modifyCmd);
            //invoke the service
            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);
} catch (Exception e) {
            e.printStackTrace();
        }
                }
```

Delete Interface Configuration

The following sample code deletes interface configuration from a specified device config.

Sample Code

```
public void testDeleteInterfaceObject() {
 try{
                                    String interfaceName = “testinterface” ;
                ModifyObjectViewRequest request = new ModifyObjectViewRequest();

      request.setAuthToken(InterfaceObjectTest.authToken);
        ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
      ObjectIdentifierType objectId = new ObjectIdentifierType();
      objectId.setCategory("deviceobj");
              objectId.setDomainId(new UnsignedInt("1"));
      ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
      objIdOrName.setObjectId(new UnsignedInt("0"));
 objIdOrName.setObjectName("SSG");
      objectId.setObjectIdOrName(objIdOrName);
      UpdateObjectViewType updateObject = new UpdateObjectViewType();
      ObjectModificationType objModType = new ObjectModificationType();
      updateObject.setObjectIdentifier(new ObjectIdentifierType[] {objectId});
      modifyCmd.setUpdateObject(updateObject);
      NodeModificationType[] deleteNode = new NodeModificationType[1];
      deleteNode[0] = new NodeModificationType();
     String strXPath = "./interface_collection/interface[.='"+interfaceName+"']";

      deleteNode[0].setDeleteNode(strXPath);
      objModType.setModification(deleteNode);
      updateObject.setObjectModification(objModType);
               modifyCmd.setUpdateObject(updateObject);
              request.addCommand(modifyCmd);
      ModifyObjectViewResponse response =
InterfaceObjectTest.stub.ModifyObjectViewRequest(request);

      assertTrue(response.getStatus() == StatusCodeType.Success);
} catch (Exception e) {
            e.printStackTrace();
        }
                }
```

Insert Gateway Configuration

The following sample code inserts Gateway configuration to the specified device config. The sample code needs **testGatewayInsert.xml** as input.

testGatewayInsert.xml

```
                              <?xml version="1.0" encoding="UTF-8"?>
<peer>
      <name_>gateway-nbitest1</name_>
      <peer-info>
        <address>
          <ip>80.2.1.1</ip>
        </address>
      </peer-info>
      <outgoing-ifp>bgroup2</outgoing-ifp>
      <preshare_rsa_dsa>
        <preshare-key>
```

```
<key>109F2U6vNuA67rsfIJC9ogwXconkOusbIDMiQ3xQ/X7fN1cuj1z1jtenT2W/Vw5tSsAPkVVDmCzS</key>

        </preshare-key>
      </preshare_rsa_dsa>
      <natt>
        <keepalive-freq>0</keepalive-freq>
      </natt>
      <modecfg>
        <server>
          <profile>N/A</profile>
        </server>
        <client>
          <update-dhcpserver>true</update-dhcpserver>
          <admin-preference>100</admin-preference>
        </client>
      </modecfg>
      <accounting />
    </peer>
```

Sample Code

```
public void testInsertGatewayObject() {
     try {
       File file = new File(webDir + File.separator + pathOfInput + "/Input/" +
"testGatewayInsert.xml");
            System.out.println("Running testInsertGatewayObject()");
            //Creating the ModifyObjectViewRequest
            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(authToken);
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("deviceobj");
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectId(new UnsignedInt("0"));
           objIdOrName.setObjectName("SSG");
            objectId.setObjectIdOrName(objIdOrName);
            objectId.setDomainId(new UnsignedInt("1"));
            //Creating a UpdateObjectViewType, this is required since the VPN is
 inserted into DeviceObj Section.
            AppendDeviceSubNodeViewType appendObject = new
AppendDeviceSubNodeViewType();
            appendObject.setObjectIdentifier(objectId);
            appendObject.setNodetype("peer");
            modifyCmd.setAppendDeviceSubNodeObject(appendObject);
            XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
            XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();
            ObjectDataType insertObjectData = new ObjectDataType();
            insertObjectData.setData(createOpaqueDataType(ome));
            appendObject.setObjecData(insertObjectData);
            request.addCommand(modifyCmd);
            //invoke the service
            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
            e.printStackTrace();
        }
}
```

Replace Gateway Configuration

The following sample code replaces existing gateway configuration for a specified device. The sample code needs **testGatewayReplace.xml** as input.

testGatewayReplace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<peer>
      <name_>test</name_>
      <peer-info>
        <address>
          <ip>80.2.1.1</ip>
        </address>
      </peer-info>
      <outgoing-ifp>bgroup1</outgoing-ifp>
      <preshare_rsa_dsa>
        <preshare-key>

<key>1O9F2U6vNuA67rsfIJC9ogwXconkOusbIDMiQ3xQ/X7fN1cuj1z1jtenT2W/Vw5tSsAPkVVDmCzS</key>

        </preshare-key>
      </preshare_rsa_dsa>
      <natt>
        <keepalive-freq>0</keepalive-freq>
      </natt>
      <modecfg>
        <server>
          <profile>N/A</profile>
        </server>
        <client>
          <update-dhcpserver>true</update-dhcpserver>
          <admin-preference>100</admin-preference>
        </client>
      </modecfg>
      <accounting />
    </peer>
```

Sample Code

```
public void testReplaceGatewayObject() {
     try {
      File file = new File(webDir + File.separator + pathOfInput + "/Input/" +
"testGatewayReplace.xml");
      System.out.println("Running testGatewayObject()");
            //Creating the ModifyObjectViewRequest
            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(authToken);
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("deviceobj");
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
           objIdOrName.setObjectId(new UnsignedInt("0"));
            objIdOrName.setObjectName("SSG");
            objectId.setObjectIdOrName(objIdOrName);
                objectId.setDomainId(new UnsignedInt("1"));
               ReplaceDeviceSubNodeViewType replaceObject = new
ReplaceDeviceSubNodeViewType();
               replaceObject.setObjectIdentifier(objectId);
            replaceObject.setNodetype("peer");
           // The gateway in NSM that has to be replaced with the XML input from
```

```
 the File testGatewayReplace.xml
            replaceObject.setNodeName(“testGateway”);
            //read the interface object in XML format from the file
"testGatewayReplace.xml"
            XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
            XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();
            ObjectDataType replaceObjectData = new ObjectDataType();
            replaceObjectData.setData(createOpaqueDataType(ome));
            replaceObject.setObjecData(replaceObjectData);
            modifyCmd.setReplaceDeviceSubNodeObject(replaceObject);
            request.addCommand(modifyCmd);
            //invoke the service
            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
          e.printStackTrace();
        }
    }
```

Delete Gateway Configuration

The following sample code deletes gateway configuration from a specified device config.

Sample Code

```
public void testDeleteGatewayObject(){
     try{
     String gatewayName = "testGateway";
      ModifyObjectViewRequest request = new ModifyObjectViewRequest();
      request.setAuthToken(GatewayObjectTest.authToken);
      ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
      ObjectIdentifierType objectId = new ObjectIdentifierType();
      objectId.setCategory("deviceobj");
             objectId.setDomainId(new UnsignedInt("1"));
      ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
      objIdOrName.setObjectId(new UnsignedInt(“0”)));
             objIdOrName.setObjectName("SSG");
      objectId.setObjectIdOrName(objIdOrName);
      UpdateObjectViewType updateObject = new UpdateObjectViewType();
      ObjectModificationType objModType = new ObjectModificationType();
      updateObject.setObjectIdentifier(new ObjectIdentifierType[] {objectId});
      modifyCmd.setUpdateObject(updateObject);
      NodeModificationType[] deleteNode = new NodeModificationType[1];
      deleteNode[0] = new NodeModificationType();
      String strXPath = "./peer_collection/peer[.='"+gatewayName+"']";
      deleteNode[0].setDeleteNode(strXPath);
      objModType.setModification(deleteNode);
      updateObject.setObjectModification(objModType);
              modifyCmd.setUpdateObject(updateObject);
              request.addCommand(modifyCmd);
       ModifyObjectViewResponse response = GatewayObjectTest
      .stub.ModifyObjectViewRequest(request);
      assertTrue(response.getStatus() == StatusCodeType.Success);
     }catch(Exception e){
       e.printStackTrace();
     }
    }
```

Insert VPN Configuration

The following sample code inserts VPN configuration to the specified device config. The sample code needs **testVPNInsert.xml** as input.

testVPNInsert

```
<?xml version="1.0" encoding="UTF-8"?>
<vpn>
<name_>vpn@nbiInsert</name_>
 <bind>
  <interface>tunnel.3</interface>
 </bind>
        <auto-manual>
          <auto>
          <gateway>gateway-nbitest1</gateway>
            <proxy>
              <ip-format_collection>
              <ip-format>
                  <local-address>
                    <v4>
                      <local-ip>10.204.77.23</local-ip>
                       <local-mask>32</local-mask>
                    </v4>
                  </local-address>
                  <remote-address>
                    <v4>
                      <remote-ip>10.204.77.24</remote-ip>
                      <remote-mask>32</remote-mask>
                    </v4>
                  </remote-address>
                  <service>any</service>
                </ip-format>
              </ip-format_collection>
            </proxy>
          </auto>
        </auto-manual>
        <backup-weight>0</backup-weight>
</vpn>
```

Sample Code

```
public void testInsertVPNObject() {
     try {
        File file =new File(webDir + File.separator + pathOfInput + "/Input/" +
"testVPNInsert.xml");
            System.out.println("Running testInsertVPNObject()");
            //Creating the ModifyObjectViewRequest
            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(authToken);
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("deviceobj");
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectId(new UnsignedInt("0"));
            objIdOrName.setObjectName("SSG");
            objectId.setObjectIdOrName(objIdOrName);
            objectId.setDomainId(new UnsignedInt("1"));
            //Creating a UpdateObjectViewType, this is required since the VPN is
 inserted into DeviceObj Section.
            AppendDeviceSubNodeViewType appendObject = new
```

```
AppendDeviceSubNodeViewType();
            appendObject.setObjectIdentifier(objectId);
            appendObject.setNodetype("vpn");
            modifyCmd.setAppendDeviceSubNodeObject(appendObject);
            XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
            XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();
            ObjectDataType vpnObjectDataType = new ObjectDataType();
            vpnObjectDataType.setData(createOpaqueDataType(ome));
            //specifying the xpath for VPN
            appendObject.setObjecData(vpnObjectDataType);
            request.addCommand(modifyCmd);
            //invoke the service
            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
          e.printStackTrace();
        }
    }
```

Replace VPN Configuration

The following sample code replaces existing VPN configuration for a specified device. The sample code needs **testVPNReplace.xml** as input.

testVPNReplace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <vpn>
<name_>vpn@nbireplace</name_>
 <bind>
  <interface>tunnel.2</interface>
 </bind>
        <auto-manual>
          <auto>
          <gateway>test</gateway>
            <proxy>
              <ip-format_collection>
              <ip-format>
                  <local-address>
                    <v4>
                      <local-ip>10.204.77.22</local-ip>
                       <local-mask>32</local-mask>
                    </v4>
                  </local-address>
                  <remote-address>
                    <v4>
                      <remote-ip>10.209.77.24</remote-ip>
                      <remote-mask>32</remote-mask>
                    </v4>
                  </remote-address>
                  <service>any</service>
                </ip-format>
              </ip-format_collection>
            </proxy>
          </auto>
        </auto-manual>
```

```
        <backup-weight>0</backup-weight>
                    </vpn>
```

Sample Code

```
public void testReplaceVPNObject() {
     try {
     File file = new File(webDir + File.separator + pathOfInput + "/Input/" +
"testVPNReplace.xml");
     System.out.println("Running testReplaceVPNObject()");
            //Creating the ModifyObjectViewRequest
            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
            request.setAuthToken(authToken);
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            ObjectIdentifierType objectId = new ObjectIdentifierType();
            objectId.setCategory("deviceobj");
            ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
            objIdOrName.setObjectId(new UnsignedInt("0"));
            objIdOrName.setObjectName("SSG");
            objectId.setObjectIdOrName(objIdOrName);
            objectId.setDomainId(new UnsignedInt("1"));
            ReplaceDeviceSubNodeViewType replaceObject = new
ReplaceDeviceSubNodeViewType();
            replaceObject.setObjectIdentifier(objectId);
            replaceObject.setNodetype("vpn");
           // The VPN in NSM that has to be replaced with the XML input from the
 File testVPNReplace.xml
            replaceObject.setNodeName(“testvpn”);
           //read the vpn object in XML format from the file "testVPNInsert.xml"

            XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
            XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();
            ObjectDataType vpnObjectDataType = new ObjectDataType();
            vpnObjectDataType.setData(createOpaqueDataType(ome));
            //specifying the xpath for VPN
           // pathValueType.setXpath("./vpn_collection/vpn");
            replaceObject.setObjecData(vpnObjectDataType);
            modifyCmd.setReplaceDeviceSubNodeObject(replaceObject);
            request.addCommand(modifyCmd);
            //invoke the service
            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);
        } catch (Exception e) {
          e.printStackTrace();
        }
}
```

Delete VPN Configuration

The following sample code deletes VPN configuration from a specified device config

Sample Code

```
public void testDeleteVPNObject(){
    try{
   String vpnName = "testvpn";
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
```

```
   request.setAuthToken(VPNObjectTest.authToken);
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
   objectId.setDomainId(new UnsignedInt("1"));
   ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
   objIdOrName.setObjectId(new UnsignedInt("0"));
   objIdOrName.setObjectName("SSG");
   objectId.setObjectIdOrName(objIdOrName);
   UpdateObjectViewType updateObject = new UpdateObjectViewType();
   ObjectModificationType objModType = new ObjectModificationType();
   updateObject.setObjectIdentifier(new ObjectIdentifierType[] {objectId});
   modifyCmd.setUpdateObject(updateObject);
   NodeModificationType[] vpnNode = new NodeModificationType[1];
   vpnNode[0] = new NodeModificationType();
   String strXPath = "./vpn_collection/vpn[.='"+vpnName+"']";
   vpnNode[0].setDeleteNode(strXPath);
   objModType.setModification(vpnNode);
   updateObject.setObjectModification(objModType);
   modifyCmd.setUpdateObject(updateObject);
   request.addCommand(modifyCmd);
 ModifyObjectViewResponse response =
VPNObjectTest.stub.ModifyObjectViewRequest(request);
    assertTrue(response.getStatus() == StatusCodeType.Success);
   }catch(Exception e){
    e.printStackTrace();
   }
    }
```

Insert Route Configuration

The following sample code inserts route configuration to the specified device config. The sample code needs **testRouteInsert.xml** as input.

testRouteInsert.xml

```
                                    <?xml version="1.0" encoding="UTF-8"?>
 <route>
            <name_ />
            <ip-address>121.23.2.0</ip-address>
            <netmask>24</netmask>
            <nexthop>
              <gateway>
                <interface>tunnel.1</interface>
                <gatewayip>10.0.0.1</gatewayip>
              </gateway>
            </nexthop>
            <preference>20</preference>
          </route>
```

Sample Code

```
public void testInsertRouteObject(){
try{
              System.out.println("Running testInserRouteObject()");
File file = new File(webDir + File.separator + pathOfInput + "/Input/" +
"testRouteInsert.xml");
ModifyObjectViewRequest request = new ModifyObjectViewRequest();
 request.setAuthToken(authToken);
 ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
              ObjectIdentifierType objectId = new ObjectIdentifierType();
```

```
                objectId.setCategory("deviceobj");
               ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
               objIdOrName.setObjectId(new UnsignedInt("0"));
               objIdOrName.setObjectName("SSG");
               objectId.setObjectIdOrName(objIdOrName);
               objectId.setDomainId(new UnsignedInt("1"));
            AppendRouteSubNodeViewType appendObject = new
AppendRouteSubNodeViewType();
            appendObject.setObjectIdentifier(objectId);
            appendObject.setVroutername(“testVrouterName”);
            XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
             XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();
            ObjectDataType natObjectData = new ObjectDataType();
             natObjectData.setData(createOpaqueDataType(ome));
             appendObject.setObjecData(natObjectData);
             modifyCmd.setAppendRouteSubNodeObject(appendObject);
             request.addCommand(modifyCmd);
     //invoke the service
       ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);

         assertTrue(response.getStatus() == StatusCodeType.Success);
 }catch (Exception e) {
  e.printStackTrace();
 }
}
```

Replace Route Configuration

The following sample code replaces existing route configuration for a specified device. The sample code needs **testRouteReplace.xml** as input.

testRouteReplace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
             <route>
            <name_ />
            <ip-address>121.23.1.0</ip-address>
            <netmask>24</netmask>
            <nexthop>
              <gateway>
                <interface>ethernet1/1</interface>
                <gatewayip>10.23.23.1</gatewayip>
              </gateway>
            </nexthop>
            <preference>20</preference>
                                  </route>
```

Sample Code

```
public void testReplaceRouteObject(){
 try{
          System.out.println("Running testReplaceRouteObject()");
          File file = new File(webDir + File.separator + pathOfInput + "/Input/"
 + "testReplaceRoute.xml");
           ModifyObjectViewRequest request = new ModifyObjectViewRequest();
```

```
            request.setAuthToken(authToken);
            ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
             ObjectIdentifierType objectId = new ObjectIdentifierType();
             objectId.setCategory("deviceobj");
             ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
             objIdOrName.setObjectId(new UnsignedInt("0"));
             objIdOrName.setObjectName("SSG");
            objectId.setObjectIdOrName(objIdOrName);
            objectId.setDomainId(new UnsignedInt("1"));
            ReplaceRouteSubNodeViewType replaceObject = new
ReplaceRouteSubNodeViewType();
            replaceObject.setObjectIdentifier(objectId);
               replaceObject.setVroutername("testVrouterName");
               replaceObject.setRoutename("testRoute");
                XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
              XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
 FileInputStream(file));
               StAXOMBuilder builder = new StAXOMBuilder(parser);
               OMElement ome = builder.getDocumentElement();
               ObjectDataType natObjectData = new ObjectDataType();
               natObjectData.setData(createOpaqueDataType(ome));
               replaceObject.setObjecData(natObjectData);
               modifyCmd.setReplaceRouteSubNodeObject(replaceObject);
               request.addCommand(modifyCmd);
         //invoke the service
            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
            assertTrue(response.getStatus() == StatusCodeType.Success);
    }catch (Exception e) {
     e.printStackTrace();
    }
    }
```

Delete Route Configuration

The following sample code deletes route configuration from a specified device config.

Sample Code

```
public void testDeleteRouteObject(){
     try{
           String routeName = "testroute";
String vrouterName = "testVrouter";
            System.out.println("Running testDeleteRouteObject()");
            ModifyObjectViewRequest request = new ModifyObjectViewRequest();
          request.setAuthToken(RouteObjectTest.authToken);
          ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
          ObjectIdentifierType objectId = new ObjectIdentifierType();
          objectId.setCategory("deviceobj");
            objectId.setDomainId(new UnsignedInt("1"));
          ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
          objIdOrName.setObjectId(new UnsignedInt("0"));
            objIdOrName.setObjectName("SSG");
          objectId.setObjectIdOrName(objIdOrName);
          UpdateObjectViewType updateObject = new UpdateObjectViewType();
          ObjectModificationType objModType = new ObjectModificationType();
         updateObject.setObjectIdentifier(new ObjectIdentifierType[] {objectId});

          modifyCmd.setUpdateObject(updateObject);
          NodeModificationType[] deleteNode = new NodeModificationType[1];
          deleteNode[0] = new NodeModificationType();
```

```
          String strXPath =
"./vrouter/vrouter_list[.='"+vrouterName+"']/route[.='"+routeName+"']";
          deleteNode[0].setDeleteNode(strXPath);
          objModType.setModification(deleteNode);
          updateObject.setObjectModification(objModType);
            modifyCmd.setUpdateObject(updateObject);
            request.addCommand(modifyCmd);
          ModifyObjectViewResponse response = RouteObjectTest
          .stub.ModifyObjectViewRequest(request);
           assertTrue(response.getStatus() == StatusCodeType.Success);
      }catch (Exception e) {
        e.printStackTrace();
      }
     }
```

Manage VPN Configuration in a junos-es Container for Junos OS Devices

The following Data Centric Service API sample codes manage a VPN configuration in a junos-es container. A junos-es container is a subcontainer of a deviceObj container. Therefore, API client code needs to be categorized as "deviceObj".

- Insert Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration on page 143
- Delete Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration on page 144
- Replace Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration on page 145

Insert Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration

testJunosVPNInsert

```
<?xml version="1.0" encoding="UTF-8"?>
         <vpn>
            <name_>testvpncomplete</name_>
            <name>testvpncomplete</name>
            <comment>test</comment>
            <df-bit>clear</df-bit>
            <vpn-monitor>
              <comment>testmonitor</comment>
              <optimized>true</optimized>
              <source-interface>1.2.2.2</source-interface>
              <destination-ip>1.2.2.2</destination-ip>
            </vpn-monitor>
            <ike>
              <comment>testike</comment>
              <gateway>vpn70-abc</gateway>
              <idle-time>60</idle-time>
              <no-anti-replay>true</no-anti-replay>
              <proxy-identity>
                <comment>testproxy</comment>
                <local>1.1.1.1/32</local>
                <remote>1::2/64</remote>
                <service>any</service>
              </proxy-identity>
              <ipsec-policy>vpn-policy1</ipsec-policy>
              <install-interval>10</install-interval>
```

```
        </ike>
        <traffic-selector>
          <name_>testtraff</name_>
          <name>testtraff</name>
          <comment>testtr</comment>
          <local-ip>1.2.2.2</local-ip>
          <remote-ip>3.3.3.3</remote-ip>
        </traffic-selector>
        <establish-tunnels>immediately</establish-tunnels>
      </vpn>
```

Sample Code

```
public  void testinsertJunosNode() {
   System.out.println("Running testInsert"+ VPN +"JunosObject()");
File file = new File(webDir + File.separator + pathOfInput +
"/Input/"+"testJunosVPNInsert.xml");
   //Create Object Identifier
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
objectId.setDomainId(new UnsignedInt("1"));
   ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
    objIdOrName.setObjectName("SRX");
   objectId.setObjectIdOrName(objIdOrName);

   //create an object of ModifyObjectViewRequest
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
   request.setAuthToken(JunosConfigTest.authToken);
   // read the new user data in XML format from the file
   XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
                                XMLStreamReader parser =
xmlInputFactory.createXMLStreamReader(new FileInputStream(file)));
   StAXOMBuilder builder = new StAXOMBuilder(parser);
   OMElement ome = builder.getDocumentElement();
   //create an object of AppendDeviceSubNodeViewType
   AppendDeviceSubNodeViewType appendObject = new AppendDeviceSubNodeViewType();

   appendObject.setObjectIdentifier(objectId);
   appendObject.setObjecData(new ObjectDataType());
   appendObject.getObjecData().setData(this.createOpaqueDataType(ome));
   appendObject.setNodetype("vpn");
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   modifyCmd.setAppendDeviceSubNodeObject(appendObject);
   request.addCommand(modifyCmd);
   // invoke the service
       ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);

        print(response);
   assertTrue(response.getStatus() == StatusCodeType.Success);
   }
```

Delete Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration

Sample Code

```
public  void testdeleteJunosNode() {
  String strXPath = null;
String deleteNode = "vpn01";
String nodeType = "vpn";

System.out.println("Running testdelete VPN JunosObject()");
```

```
//Create Object Identifier
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
objectId.setDomainId(new UnsignedInt("1"));
   ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
    objIdOrName.setObjectName("SRX");
   objectId.setObjectIdOrName(objIdOrName);

               //create an object of ModifyObjectViewRequest
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
   request.setAuthToken(JunosConfigTest.authToken);
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   UpdateObjectViewType updateObject = new UpdateObjectViewType();
   updateObject.setObjectIdentifier(new ObjectIdentifierType[]{objectId});

   modifyCmd.setUpdateObject(updateObject);
   NodeModificationType nodeModificationType = new NodeModificationType();
   if(nodeType.equalsIgnoreCase("vpn")){
    strXPath = "./configuration/security/ipsec/vpn[.='"+deleteNode+"']";
   }else if(nodeType.equalsIgnoreCase("gateway")){
    strXPath = "./configuration/security/ike/gateway[.='"+deleteNode+"']";;
   }else if(nodeType.equalsIgnoreCase("static_route")){
    strXPath = "./configuration/routing-options/static/route[.='"+deleteNode+"']";

   }else if(nodeType.equalsIgnoreCase("ike_policy")){
    strXPath = "./configuration/security/ike/policy[.='"+deleteNode+"']";
   }else if(nodeType.equalsIgnoreCase("ipsec_policy")){
    strXPath = "./configuration/security/ipsec/policy[.='"+deleteNode+"']";
   }else if(nodeType.equalsIgnoreCase("interface")){
    strXPath = "./configuration/interfaces/interface[.='"+deleteNode+"']";
   }
   nodeModificationType.setDeleteNode(strXPath);
   SubObjectModificationType subObjModificationType = new
SubObjectModificationType();
   subObjModificationType.setSubCategory("junos-es");
   subObjModificationType.addModification(nodeModificationType);
   updateObject.setObjectModification(new ObjectModificationType());

updateObject.getObjectModification().addSubObjectModification(subObjModificationType);

   request.addCommand(modifyCmd);
   //invoke service
   ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
print(response);
   assertTrue(response.getStatus() == StatusCodeType.Success);
```

Replace Interface/gateway/static_route/ike_policy/ipsec_policy/vpn Configuration

testJunosVPNReplace

```
<?xml version="1.0" encoding="UTF-8"?>
         <vpn>
            <name_>testvpnreplace</name_>
            <name>testvpncomplete</name>
            <comment>test</comment>
            <df-bit>clear</df-bit>
            <vpn-monitor>
              <comment>testmonitor</comment>
              <optimized>true</optimized>
              <source-interface>1.2.2.2</source-interface>
              <destination-ip>1.2.2.2</destination-ip>
```

```
    </vpn-monitor>
    <ike>
      <comment>testike</comment>
      <gateway>vpn70-abc</gateway>
      <idle-time>60</idle-time>
      <no-anti-replay>true</no-anti-replay>
      <proxy-identity>
        <comment>testproxy</comment>
        <local>1.1.1.1/32</local>
        <remote>1::2/64</remote>
        <service>any</service>
      </proxy-identity>
      <ipsec-policy>vpn-policy1</ipsec-policy>
      <install-interval>10</install-interval>
    </ike>
    <traffic-selector>
      <name_>testtraff</name_>
      <name>testtraff</name>
      <comment>testtr</comment>
      <local-ip>1.2.2.2</local-ip>
      <remote-ip>3.3.3.3</remote-ip>
    </traffic-selector>
    <establish-tunnels>immediately</establish-tunnels>
  </vpn>
```

Sample Code

```
                                                    public  void testReplaceJunosNode()
 {
String replaceNode = “testvpncomplete”;
   System.out.println("Running tesreplace"+ VPN +"JunosObject()");
File file = new File(webDir + File.separator + pathOfInput +
"/Input/"+"testJunosVPNreplace.xml");
   //Create Object Identifier
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
objectId.setDomainId(new UnsignedInt(“1”));
   ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
    objIdOrName.setObjectName(“SRX”);
   objectId.setObjectIdOrName(objIdOrName);

   //create an object of ModifyObjectViewRequest
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
   request.setAuthToken(JunosConfigTest.authToken);

   // read the user data in XML format from the file
   XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
           XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
   StAXOMBuilder builder = new StAXOMBuilder(parser);
   OMElement ome = builder.getDocumentElement();

   //create an object of ReplaceDeviceSubNodeViewType
  ReplaceDeviceSubNodeViewType replaceObject = new ReplaceDeviceSubNodeViewType();

   replaceObject.setObjectIdentifier(objectId);
   replaceObject.setObjecData(new ObjectDataType());
   replaceObject.getObjecData().setData(this.createOpaqueDataType(ome));

   if(replaceNode!= null){
    replaceObject.setNodeName(replaceNode);
```

```
   }else{
    throw new Exception("replace value is null!!! " );
   }
   replaceObject.setNodetype("vpn");

   //create an object of  ModifyViewCommandType
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   modifyCmd.setReplaceDeviceSubNodeObject(replaceObject);
   request.addCommand(modifyCmd);
   // invoke the service
   ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
print(response);
   assertTrue(response.getStatus() == StatusCodeType.Success);
   }
```

Manage NAT Configuration in DeviceObj Container

This section describes the data centric service API sample codes that can be used to manage NAT configuration in deviceObj.

- Insert NAT (MIP/DIP/VIP) Configuration on page 147
- Replace NAT (MIP/DIP/VIP) configuration on page 149
- Delete NAT (MIP/DIP/VIP) Configuration on page 151

Insert NAT (MIP/DIP/VIP) Configuration

The following sample code inserts NAT (MIP/VIP/DIP) configuration to the specified device config. The sample code needs **testMIPInsert.xml**, **testVIPInsert.xml**, and **testDIPInsert.xml** as input and the same sample code can be used for MIP/VIP also.

testNATMIPInsert.xml

```
 <?xml version="1.0" encoding="UTF-8"?>
  <mip>
          <name_>1.1.1.1</name_>
          <host>2.2.2.2</host>
          <vrouter>trust-vr</vrouter>
        </mip>
 <mip>
          <name_>3333::</name_>
          <isIpv6>true</isIpv6>
          <netmask>32</netmask>
          <htype>ipv6-network</htype>
          <hip>
            <v6>
              <ip>3333:3333:2222:444::</ip>
              <prefix>64</prefix>
            </v6>
   </hip> <vrouter>trust-vr</vrouter> </mip>
```

testNATVIPInsert.xml

```
<?xml version="1.0" encoding="UTF-8"?>
                                      <vipset>
                                      <name_>140.1.1.20</name_>
                                      <mappings_collection>
                                       <mappings>
<name_>2048</name_>
```

```
<map_ip>2.2.2.22</map_ip>
<service>&1.service.492</service>
     <manual>true</manual>
     </mappings>
</mappings_collection>
                                          </vipset>
```

testNATDIPInsert.xml

```
<?xml version="1.0" encoding="UTF-8"?>
      <dip>
          <name_>8</name_>
          <dtype>ipv6-range</dtype>
          <shift-from />
          <ip1 />
          <ip-min>
            <v6>
              <ip>8888:8888:8888:8888:8888:8888:8888:8886</ip>
            </v6>
          </ip-min>
          <ip2 />
          <ip-max>
            <v6>
              <ip>8888:8888:8888:8888:8888:8888:8888:8989</ip>
            </v6>
          </ip-max>
          <scale-size>3</scale-size>
          <xtype>ipv6</xtype>
          <vx_extended_ip>
            <v6>
              <ip>8888:8888:8888:8888:8888:8888:8888:0</ip>
              <prefix>96</prefix>
            </v6>
          </vx_extended_ip>
                                 </dip>
```

Sample Code

```
public void testInsertNatDipObject() {
     try{
 ModifyObjectViewRequest request = new ModifyObjectViewRequest();
 request.setAuthToken(authToken);
 File file = new File(webDir + File.separator + pathOfInput + "/Input/" +
"testNATDipInsert.xml");
                System.out.println("Running testInsertNATDipObject()");
                ObjectIdentifierType objectId = new ObjectIdentifierType();
                objectId.setCategory("deviceobj");
           ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
          objIdOrName.setObjectId(new UnsignedInt("0"));
                objIdOrName.setObjectName("SSG");
             objectId.setObjectIdOrName(objIdOrName);
            objectId.setDomainId(new UnsignedInt("1"));
           ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
            NATNodeType natNode = new NATNodeType();
            //For MIP
           // natNode.setNodetype(“mip”);
           // For DIP
              natNode.setNodetype(“dip”);
           // For VIP
          //  natNode.setNodetype(“vip”);
            natNode.setInterfacename(“ethernet0/0”);
          AppendNATSubNodeViewType appendObject = new AppendNATSubNodeViewType();
```

```
            appendObject.setObjectIdentifier(objectId);
            appendObject.setNatobject(natNode);
            XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
            StAXOMBuilder builder = new StAXOMBuilder(parser);
            OMElement ome = builder.getDocumentElement();
            ObjectDataType natObjectData = new ObjectDataType();
            natObjectData.setData(createOpaqueDataType(ome));
            appendObject.setObjecData(natObjectData);
            modifyCmd.setAppendNATSubNodeObject(appendObject);
            request.addCommand(modifyCmd);
            //invoke the service
            ModifyObjectViewResponse response =
stub.ModifyObjectViewRequest(request);
           assertTrue(response.getStatus() == StatusCodeType.Success);
    }catch (Exception e) {
      e.printStackTrace();
}
 }
```

Replace NAT (MIP/DIP/VIP) configuration

The following sample code replaces the existing NAT (MIP/VIP/DIP) configuration for a specified device. The sample code needs **testMIPReplace.xml**, **testVIPReplace.xml**, and **testDIPReplace.xml** as input and the same sample code can be used for MIP/VIP also.

testNATMIPReplace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <mip>
          <name_>1.1.1.1</name_>
          <host>2.2.2.2</host>
          <vrouter>trust-vr</vrouter>
        </mip>
 <mip>
          <name_>3333::</name_>
          <isIpv6>true</isIpv6>
          <netmask>32</netmask>
          <htype>ipv6-network</htype>
          <hip>
            <v6>
              <ip>3333:3333:2222:444::</ip>
              <prefix>64</prefix>
            </v6>
                        </hip> <vrouter>trust-vr</vrouter> </mip>
```

testNATVIPReplace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
                                                                <vipset>

<name_>140.1.1.20</name_>

<mappings_collection>
                                                                                <mappings>
<name_>2048</name_>
<map_ip>2.2.2.22</map_ip>
<service>&1.service.492</service>
```

```
     <manual>true</manual>
     </mappings>
</mappings_collection>
         </vipset>
```

testNATDIPReplace.xml

```
<?xml version="1.0" encoding="UTF-8"?>
      <dip>
          <name_>8</name_>
          <dtype>ipv6-range</dtype>
          <shift-from />
          <ip1 />
          <ip-min>
            <v6>
              <ip>8888:8888:8888:8888:8888:8888:8888:8886</ip>
            </v6>
          </ip-min>
          <ip2 />
          <ip-max>
            <v6>
              <ip>8888:8888:8888:8888:8888:8888:8888:8989</ip>
            </v6>
          </ip-max>
          <scale-size>3</scale-size>
          <xtype>ipv6</xtype>
          <vx_extended_ip>
            <v6>
              <ip>8888:8888:8888:8888:8888:8888:8888:0</ip>
              <prefix>96</prefix>
            </v6>
          </vx_extended_ip>
                                  </dip>
```

Sample Code

```
public void testReplaceNatDipObject(){
 try{
   System.out.println("Running testReplaceNatDipObject()");
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
    request.setAuthToken(authToken);
    File file =  new File(webDir + File.separator + pathOfInput + "/Input/" +
"testNATDipReplace.xml");
    ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
    ObjectIdentifierType objectId = new ObjectIdentifierType();
    objectId.setCategory("deviceobj");
    ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
    objIdOrName.setObjectId(new UnsignedInt("0"));
    objIdOrName.setObjectName("SSG");
    objectId.setObjectIdOrName(objIdOrName);
    objectId.setDomainId(new UnsignedInt("1"));
    NATNodeType natNode = new NATNodeType();
    //For DIP
    natNode.setNodetype("dip");
    //For VIP
     // natNode.setNodetype("vip");
    //For MIP
    // natNode.setNodetype("mip");
    natNode.setInterfacename(“ethernet0/0”);
    ReplaceNATSubNodeViewType replaceObject = new ReplaceNATSubNodeViewType();
```

```
    replaceObject.setObjectIdentifier(objectId);
    replaceObject.setNatobject(natNode);
    replaceObject.setNatnodename("4");
   XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
    XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
    StAXOMBuilder builder = new StAXOMBuilder(parser);
    OMElement ome = builder.getDocumentElement();
    ObjectDataType natObjectData = new ObjectDataType();
    natObjectData.setData(createOpaqueDataType(ome));
    replaceObject.setObjecData(natObjectData);
    modifyCmd.setReplaceNATSubNodeObject(replaceObject);
    request.addCommand(modifyCmd);
    //invoke the service
     ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
      assertTrue(response.getStatus() == StatusCodeType.Success);
 }catch (Exception e) {
       e.printStackTrace();
     }
    }
```

Delete NAT (MIP/DIP/VIP) Configuration

The following sample code deletes NAT (MIP/DIP/VIP) configuration from specified device config. The same sample code can be used for MIP and VIP.

Sample Code

```
public void testDeleteNATDipObject(){
  try{
String deleteNodeName = "8";
String interfaceName = "ethernet0/0";
//String nodeType = "mip"; //For MIP
//String nodeType = "vip"; //For VIP
String nodeType = "dip"; //For DIP
String interfaceName = "ethernet1/4"; //For DIP
ModifyObjectViewRequest request = new ModifyObjectViewRequest();
request.setAuthToken(NATObjectTest.authToken);
ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
ObjectIdentifierType objectId = new ObjectIdentifierType();
objectId.setCategory("deviceobj");
 objectId.setDomainId(new UnsignedInt("1"));
ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
objIdOrName.setObjectId(new UnsignedInt("0"));
 objIdOrName.setObjectName("SSG");
objectId.setObjectIdOrName(objIdOrName);
UpdateObjectViewType updateObject = new UpdateObjectViewType();
ObjectModificationType objModType = new ObjectModificationType();
updateObject.setObjectIdentifier(new ObjectIdentifierType[] {objectId});
modifyCmd.setUpdateObject(updateObject);
NodeModificationType[] deleteNode = new NodeModificationType[1];
deleteNode[0] = new NodeModificationType();
String strXPath =
"./interface_collection/interface[.='"+interfaceName+"']/"+nodeType+"[.='"+deleteNodeName+"']";

 deleteNode[0].setDeleteNode(strXPath);
objModType.setModification(deleteNode);
updateObject.setObjectModification(objModType);
  modifyCmd.setUpdateObject(updateObject);
request.addCommand(modifyCmd);
ModifyObjectViewResponse response =
```

```
NATObjectTest.stub.ModifyObjectViewRequest(request);
assertTrue(response.getStatus() == StatusCodeType.Success);
}catch(Exception e){
      e.printStackTrace();
}
  }
```

Admin User Management

This section describes the data centric service API sample code that manages user configuration in deviceObj.

- Insert Admin Configuration on page 152
- Replace Admin Configuration on page 153
- Delete Admin Configuration on page 154

Insert Admin Configuration

The following sample code inserts user configuration to the specified device config. The password is encrypted in the Server. The sample code needs **testInsertUser.xml** as input.

testInsertUser.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <user>
  <name_>user1</name_>
  <password>netscreen123</password>
  <privilege>root</privilege>
  <trustee />
  </user>
```

Sample Code

```
public void testAppendNewUserObj() {
 try {
 System.out.println("Running testInsertUserObject()");

  //create an object of ModifyObjectViewRequest
 ModifyObjectViewRequest request = new ModifyObjectViewRequest();
 request.setAuthToken(UserManagementTest.authToken);
 File file = new File(webDir + File.separator + pathOfInput + "/Input/"
+"testInsertUser.xml");

 //Create Object Identifier
              ObjectIdentifierType objectId = new ObjectIdentifierType();
                  objectId.setCategory("deviceobj");
                  ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
                  objIdOrName.setObjectId(new UnsignedInt("0"));
                  objIdOrName.setObjectName("SSG");
                  objectId.setObjectIdOrName(objIdOrName);
                  objectId.setDomainId(new UnsignedInt("1"));
          // read the new user data in XML format from the file
 XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
 XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file.getAbsolutePath()));
 StAXOMBuilder builder = new StAXOMBuilder(parser);
 OMElement ome = builder.getDocumentElement();
```

```
 //create an object of AppendDeviceSubNodeViewType
AppendDeviceSubNodeViewType appendObject = new AppendDeviceSubNodeViewType();
appendObject.setObjectIdentifier(objectId);
appendObject.setObjecData(new ObjectDataType());
appendObject.getObjecData().setData(this.createOpaqueDataType(ome));
                appendObject.setNodetype("admin.user");
 ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
modifyCmd.setAppendDeviceSubNodeObject(appendObject);
request.addCommand(modifyCmd);
    //invoke the service
ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
assertTrue(response.getStatus() == StatusCodeType.Success);
} catch (Exception e) {
  e.printStackTrace();
}
}
```

Replace Admin Configuration

The following sample code replaces the existing user configuration to the specified device config. The password is encrypted in the Server. The sample code needs **testReplaceUser.xml** as input.

testReplaceUser.xml

```
                                        <?xml version="1.0" encoding="UTF-8"?>
 <user>
 <name_>user2</name_>
 <password>netscreen12356</password>
 <privilege>root</privilege>
 <trustee />
 </user>
```

Sample Code

```
 public void testReplaceUserObject() {
  try {
  System.out.println("Running testReplaceUserObject()");
  File file = new File(webDir + File.separator + pathOfInput + "/Input/" +
"testReplaceUser.xml");
   //create an object of ModifyObjectViewRequest
  ModifyObjectViewRequest request = new ModifyObjectViewRequest();
  request.setAuthToken(UserManagementTest.authToken);
  //Create Object Identifier
             ObjectIdentifierType objectId = new ObjectIdentifierType();
                 objectId.setCategory("deviceobj");
                 ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
                                 objIdOrName.setObjectId(new UnsignedInt("0"));

                 objIdOrName.setObjectName("SSG");
                 objectId.setObjectIdOrName(objIdOrName);
                 objectId.setDomainId(new UnsignedInt("1"));
  // read the user data in XML format from the file
  XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file.getAbsolutePath()));
  StAXOMBuilder builder = new StAXOMBuilder(parser);
  OMElement ome = builder.getDocumentElement();
  //create an object of ReplaceDeviceSubNodeViewType
  ReplaceDeviceSubNodeViewType replaceObject = new ReplaceDeviceSubNodeViewType();
```

```
 replaceObject.setObjectIdentifier(objectId);
 replaceObject.setObjecData(new ObjectDataType());
 replaceObject.getObjecData().setData(this.createOpaqueDataType(ome));
  replaceObject.setNodeName("User1");
  replaceObject.setNodetype("admin.user");
  //create an object of  ModifyViewCommandType
 ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
 modifyCmd.setReplaceDeviceSubNodeObject(replaceObject);
 request.addCommand(modifyCmd);
 //invoke Service
 ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
 assertTrue(response.getStatus() == StatusCodeType.Success);
 } catch (Exception e) {
    e.printStackTrace();
 }
}
```

Delete Admin Configuration

The following sample code deletes admin configuration from specified device config.

Sample Code

```
                                        public void testDeleteUserObject() {
 try {
      String deleteUser = "User1";
      System.out.println("Running testdeleteUserObject()");
      //Create Object Identifier
      ObjectIdentifierType objectId = new ObjectIdentifierType();
       objectId.setCategory("deviceobj");
       ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
                        objIdOrName.setObjectId(new UnsignedInt("0"));
        objIdOrName.setObjectName("SSG");
        objectId.setObjectIdOrName(objIdOrName);
        objectId.setDomainId(new UnsignedInt("1"));
       //create an object of ModifyObjectViewRequest
       ModifyObjectViewRequest request = new ModifyObjectViewRequest();
       request.setAuthToken(UserManagementTest.authToken);
       //create an object of UpdateObjectViewType
       UpdateObjectViewType updateObject = new UpdateObjectViewType();
      updateObject.setObjectIdentifier(new ObjectIdentifierType[] { objectId });

       PathValueType pathValueType = new PathValueType();
       pathValueType.setXpath("./admin/user_collection/user[.='"+ deleteUser +
"']");
       NodeModificationType userNode = new NodeModificationType();
       userNode.setDeleteNode(pathValueType.getXpath().toString());
       updateObject.setObjectModification(new ObjectModificationType());
       updateObject.getObjectModification().addModification(userNode);
        //create an object of ModifyViewCommandType
       ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
       modifyCmd.setUpdateObject(updateObject);
      request.addCommand(modifyCmd);
       //invoke the service
      ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);

       assertTrue(response.getStatus() == StatusCodeType.Success);
  } catch (Exception e) {
  e.printStackTrace();
 }
```

```
}
```

Modification of Management interface for Junos OS Devices

The following Data Centric Service API sample codes modify the configuration of management interfaces like fxp0 and fxp1 deviceObj.

- Insert Management Interface Configuration on page 155
- Replace Management Interface Configuration on page 156
- Delete Management Interface Configuration on page 157

Insert Management Interface Configuration

testJunosMgtInterfaceInsert:

```
<?xml version="1.0" encoding="UTF-8"?>
 <interface>
      <name>fxp0</name>
      <comment>test</comment>
      <description>test</description>
      <disable>true</disable>
      <unit>
        <name>0</name>
        <family>
          <inet/>
          <inet6>
            <address>
              <name>29::1/64</name>
            </address>
          </inet6>
        </family>
      </unit>
    </interface>
```

Sample Code

```
public  void testinsertMgtInterfaceNode() {
   System.out.println("Running testInsert"+ VPN +"JunosObject()");
File file = new File(webDir + File.separator + pathOfInput +
"/Input/"+"testJunosMgtInterfaceInsert.xml");
   //Create Object Identifier
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
objectId.setDomainId(new UnsignedInt("1"));
   ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
    objIdOrName.setObjectName("SRX");
   objectId.setObjectIdOrName(objIdOrName);

   //create an object of ModifyObjectViewRequest
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
   request.setAuthToken(JunosConfigTest.authToken);
   // read the new user data in XML format from the file
   XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
       XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file)));
   StAXOMBuilder builder = new StAXOMBuilder(parser);
   OMElement ome = builder.getDocumentElement();
```

```
   //create an object of AppendDeviceSubNodeViewType
   AppendDeviceSubNodeViewType appendObject = new AppendDeviceSubNodeViewType();

   appendObject.setObjectIdentifier(objectId);
   appendObject.setObjecData(new ObjectDataType());
   appendObject.getObjecData().setData(this.createOpaqueDataType(ome));
   appendObject.setNodetype("interface");

appendObject.setGroupName("node0");
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   modifyCmd.setAppendDeviceSubNodeObject(appendObject);
   request.addCommand(modifyCmd);
   // invoke the service
   ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
print(response);
   assertTrue(response.getStatus() == StatusCodeType.Success);
   }
```

Replace Management Interface Configuration

testJunosMgtInterfacereplace:

```
<?xml version="1.0" encoding="UTF-8"?>
 <interface>
      <name>fp0</name>
      <comment>test</comment>
      <description>test</description>
      <disable>true</disable>
      <unit>
        <name>1</name>
        <family>
          <inet/>
          <inet6>
            <address>
              <name>30::1/64</name>
            </address>
          </inet6>
        </family>
      </unit>
    </interface>
```

Sample Code

```
public  void testReplaceMgtinterfaceNode() {
String replaceNode = "testvpncomplete";
   System.out.println("Running tesreplace"+ VPN +"JunosObject()");
File file = new File(webDir + File.separator + pathOfInput +
"/Input/"+"testJunosVPNreplace.xml");
   //Create Object Identifier
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
objectId.setDomainId(new UnsignedInt("1"));
   ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
objIdOrName.setObjectName("SRX");
   objectId.setObjectIdOrName(objIdOrName);

   //create an object of ModifyObjectViewRequest
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
   request.setAuthToken(JunosConfigTest.authToken);

   // read the user data in XML format from the file
```

```
   XMLInputFactory xmlInputFactory = XMLInputFactory.newInstance();
           XMLStreamReader parser = xmlInputFactory.createXMLStreamReader(new
FileInputStream(file));
   StAXOMBuilder builder = new StAXOMBuilder(parser);
   OMElement ome = builder.getDocumentElement();

   //create an object of ReplaceDeviceSubNodeViewType
  ReplaceDeviceSubNodeViewType replaceObject = new ReplaceDeviceSubNodeViewType();

   replaceObject.setObjectIdentifier(objectId);
   replaceObject.setObjecData(new ObjectDataType());
   replaceObject.getObjecData().setData(this.createOpaqueDataType(ome));

   if(replaceNode!= null){
    replaceObject.setNodeName(replaceNode);
   }else{
    throw new Exception("replace value is null!!! " );
   }
   replaceObject.setNodetype("interface");
replaceObject.setGroupName("node0");
   //create an object of  ModifyViewCommandType
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   modifyCmd.setReplaceDeviceSubNodeObject(replaceObject);
   request.addCommand(modifyCmd);
   // invoke the service
   ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
print(response);
   assertTrue(response.getStatus() == StatusCodeType.Success);
   }
```

Delete Management Interface Configuration

Sample Code

```
public  void testdeleteMgtInterfaceNode() {
  String strXPath = null;
String deleteNode = "vpn01";
String nodeType = "vpn";

System.out.println("Running testdelete VPN JunosObject()");
//Create Object Identifier
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
objectId.setDomainId(new UnsignedInt("1"));
ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
    objIdOrName.setObjectName("SRX");
   objectId.setObjectIdOrName(objIdOrName);

               //create an object of ModifyObjectViewRequest
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
   request.setAuthToken(JunosConfigTest.authToken);

   DeleteMgmtIntViewType deleteObject = new  DeleteMgmtIntViewType();

   deleteObject.setObjectIdentifier(objectId);
   deleteObject.setObjecData(new ObjectDataType());
   deleteObject.setNodeName(deleteNode);
   deleteObject.setNodetype(nodeType);
   deleteObject.setGroupName(groupName);

   //create an object of  ModifyViewCommandType
```

```
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   modifyCmd.setDeleteMgmtIntObject(deleteObject);
   request.addCommand(modifyCmd);
   //invoke service
   ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
print(response);
   assertTrue(response.getStatus() == StatusCodeType.Success);
```

Assign SNMP Name and SNMP Contact for Junos OS Devices

The following Data Centric Service API sample codes assign the SNMP name and SNMP contact for Junos OS devices.

Sample Code

```
public void testUpdateSNMPNodeRequestForJunOS() {
  String domainId = null;
  String deviceName = null;
  String deviceId = null;
  String snmpName = null;
  String snmpContact = null;
  Properties props = null;
  File userInput = null;
  String missingArgs = null;
  OpaqueDataType data = null;

  System.out.println("Running SNMP Node Request for Junos");
  try{
  if(this.configInputFilePath == null)
  userInput= new File(webDir + File.separator + pathOfInput+ "/Input/SRX/" +
"testJunosConfig.txt");
   else
userInput = new File(this.configInputFilePath);
   if (null != userInput) {
    props = new Properties();
    FileInputStream fin = new FileInputStream(userInput);
    props.load(fin);
    fin.close();
   }
   // Load input from "testJunosConfig.txt" file
   domainId = props.get("deviceobj.domainId").toString();
   deviceName = props.get("deviceobj.deviceName").toString();
   deviceId = props.get("deviceobj.deviceId").toString();
   snmpName = props.get("deviceobj.junos-es.snmpName").toString();
   snmpContact = props.get("deviceobj.junos-es.snmpContact").toString();
   if( (null == deviceId || "".equals(deviceId)) && (null == deviceName ||
"".equals(deviceName)))
    missingArgs = missingArgs+
    "Device ID & Device Name is not configured in "+userInput.getName()+
    "\nPlease configure the same as 'deviceobj.deviceId' or deviceobj.deviceName
 \n";
   if(null == domainId || "".equals(domainId))
    missingArgs = missingArgs+
    "Domain Id is not configured in " +userInput.getName()+
    " Please configure the same as 'deviceobj.domainId' \n";
   if(!"".equals(missingArgs) && missingArgs != null){
    if(logger!=null){
     logger.error(missingArgs);
     logger.error("Missing Arguments Exiting....");
    }
```

```
    else{
     System.out.println(missingArgs);
     System.out.println("Missing Arguments Exiting....");
    }
    fail();
   }
   ModifyObjectViewRequest request = new ModifyObjectViewRequest();
   request.setAuthToken(JunosConfigTest.authToken);
   ModifyViewCommandType modifyCmd = new ModifyViewCommandType();
   ObjectIdentifierType objectId = new ObjectIdentifierType();
   objectId.setCategory("deviceobj");
   objectId.setDomainId(new UnsignedShort(domainId));
   ObjectIdOrNameType objIdOrName = new ObjectIdOrNameType();
   if(deviceName != null){
    objIdOrName.setObjectName(deviceName);
   }else {
    objIdOrName.setObjectId(new UnsignedInt(deviceId));
   }
   objectId.setObjectIdOrName(objIdOrName);

   UpdateObjectViewType updateObject = new UpdateObjectViewType();
   updateObject.setObjectIdentifier(new ObjectIdentifierType[]{objectId});
   modifyCmd.setUpdateObject(updateObject);
NodeModificationType nodeModificationType = new NodeModificationType();
   PathValueType pathValueType = new PathValueType();
   //nodeModificationType.setUpdateNode(pathValueType);
   nodeModificationType.setAppendNode(pathValueType);
   SubObjectModificationType subObjModificationType = new
SubObjectModificationType();
   subObjModificationType.setSubCategory("junos-es");
   subObjModificationType.addModification(nodeModificationType);
   updateObject.setObjectModification(new ObjectModificationType());

updateObject.getObjectModification().addSubObjectModification(subObjModificationType);

   if(snmpName != null) {
    pathValueType.setXpath("./configuration/snmp");
    data = this.createOpaqueDataType("<system-name>"+snmpName+"</system-name>");

   }
   if(snmpContact != null){
    pathValueType.setXpath("./configuration/snmp/contact");
    data = this.createOpaqueDataType("<name>"+snmpContact+"</name>");
   }
   pathValueType.setValue(data);
   request.addCommand(modifyCmd);
   ModifyObjectViewResponse response = stub.ModifyObjectViewRequest(request);
   print(response);
   assertTrue(response.getStatus() == StatusCodeType.Success);
  } catch (Exception e) {
   e.printStackTrace();
  }
 }
```

Limitations

- The SNMP name and contact cannot be assigned simultaneously. They must be assigned one after another.
- Only assignment of the SNMP name and contact is supported. Deletion of the SNMP name and contact is not supported.

PART 5

NSM API WSDLs

This part describes the WSDLs that define the NSM APIs.

NOTE: In these WSDL files, the expression “nbi” refers to the API.

This part contains the following chapters:

- Job Service API WSDL on page 163
- System Service API WSDL on page 173
- Data Centric API WSDL on page 179
- Log Service API WSDL on page 191

CHAPTER 15

Job Service API WSDL

This chapter describes the Job Service API WSDL.

- WSDL File on page 163

WSDL File

The WSDL definition file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 rel. 3 sp1 (http://www.altova.com) by Shaogang Chen
 (Juniper Networks, Inc.) -->
<wsdl:definitions xmlns:impl="http://juniper.net/webproxy/JobService"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:core="http://juniper.net/core"
 xmlns:os="http://juniper.net/webproxy/objectservice"
xmlns:ns="http://schemas.xmlsoap.org/soap/encoding/" name="JobService"
targetNamespace="http://juniper.net/webproxy/JobService">
 <wsdl:types>
  <xs:schema version="1.0" elementFormDefault="qualified"
targetNamespace="http://juniper.net/webproxy/JobService"
xmlns="http://juniper.net/webproxy/JobService" xmlns:core="http://juniper.net/core"
 xmlns:impl="http://juniper.net/webproxy/JobService">
   <xs:import namespace="http://juniper.net/core"
schemaLocation="common/BaseMessages.xsd"/>
   <xs:complexType name="JobArgsType">
    <xs:annotation>
     <xs:documentation>The list of devices that the job is applied to:
     domainName: the name of the domain
     domainId: the id of the domain
     deviceId: the list of the device id
                        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
     <xs:element name="domainId" type="xs:unsignedInt"/>
     <xs:element name="deviceId" type="xs:unsignedInt" maxOccurs="unbounded"/>
    </xs:sequence>
   </xs:complexType>
   <xs:complexType name="JobRequestType">
    <xs:annotation>
     <xs:documentation>The common parameters of the job request
```

```
jobName: the job name
scheduleTime: the time when the job is expected to run. If not specified, the job
 shall run immediately
jobArgs: the list of the devices that the job is applied to
                        </xs:documentation>
    </xs:annotation>
    <xs:sequence>
     <xs:element name="jobName" type="xs:string" minOccurs="0"/>
     <xs:element name="scheduleTime" type="xs:date" minOccurs="0"/>
     <xs:element name="jobArgs" type="impl:JobArgsType"/>
    </xs:sequence>
   </xs:complexType>
   <xs:simpleType name="JobStatusType">
    <xs:restriction base="xs:token">
     <xs:enumeration value="STARTED"/>
     <xs:enumeration value="STOPPED"/>
     <xs:enumeration value="CANCELLED"/>
     <xs:enumeration value="INPROGRESS"/>
     <xs:enumeration value="QUEUED"/>
     <xs:enumeration value="COMPLETEDWITHSUCCESS"/>
     <xs:enumeration value="COMPLETEDWITHFAILURE"/>
     <xs:enumeration value="ERROR"/>
     <xs:enumeration value="DEVSVRDOWN"/>
     <xs:enumeration value="DEVICE_LIMIT"/>
     <xs:enumeration value="UNKNOWN"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="JobResponseType">
    <xs:sequence>
     <xs:element name="status" type="impl:JobStatusType"/>
     <xs:element name="jobName" type="xs:string"/>
     <xs:element name="explanation" type="xs:string" minOccurs="0"/>
    </xs:sequence>
   </xs:complexType>
   <xs:element name="UpdateDeviceRequest">
    <xs:annotation>
     <xs:documentation>Update the device configuration
                        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleRequestType">
       <xs:sequence>
        <xs:element name="jobRequest" type="impl:JobRequestType"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="UpdateDeviceResponse">
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleResponseType">
       <xs:sequence>
        <xs:element name="jobResponse" type="impl:JobResponseType"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="ImportDeviceRequest">
```

```
    <xs:annotation>
     <xs:documentation>Import the device configuration from the physical devices

                        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleRequestType">
       <xs:sequence>
        <xs:element name="jobRequest" type="impl:JobRequestType"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="ImportDeviceResponse">
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleResponseType">
       <xs:sequence>
        <xs:element name="jobResponse" type="impl:JobResponseType"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
<xs:element name="ValidateDeviceRequest">
<xs:annotation>
<xs:documentation>Validate the device configuration from the physical devices
</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:complexContent>
<xs:extension base="core:SimpleRequestType">
<xs:sequence>
<xs:element name="jobRequest" type="impl:JobRequestType"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="ValidateDeviceResponse">
<xs:complexType>
<xs:complexContent>
<xs:extension base="core:SimpleResponseType">
<xs:sequence>
<xs:element name="jobResponse" type="impl:JobResponseType"/>
</xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:element>
   <xs:element name="GetConfigSummaryRequest">
    <xs:annotation>
     <xs:documentation>Get the modeled device configuration summarization to be
sent to the managed device during the next device update.
                        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleRequestType">
```

```
      <xs:sequence>
       <xs:element name="jobRequest" type="impl:JobRequestType"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetConfigSummaryResponse">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleResponseType">
      <xs:sequence>
       <xs:element name="jobResponse" type="impl:JobResponseType"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetRunningConfigRequest">
   <xs:annotation>
    <xs:documentation>Get the device configuration summarization currently running
on an actual physical device
                        </xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleRequestType">
      <xs:sequence>
       <xs:element name="jobRequest" type="impl:JobRequestType"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetRunningConfigResponse">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleResponseType">
      <xs:sequence>
       <xs:element name="jobResponse" type="impl:JobResponseType"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="CancelJobRequest">
   <xs:annotation>
    <xs:documentation>Cancel a running job
                        </xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleRequestType">
      <xs:sequence>
       <xs:element name="domainId" type="xs:unsignedInt"/>
       <xs:element name="jobName" type="xs:string"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
```

```
  <xs:element name="CancelJobResponse">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleResponseType">
      <xs:sequence>
       <xs:element name="jobResponse" type="impl:JobResponseType"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetDeltaConfigRequest">
   <xs:annotation>
    <xs:documentation>Get the differences between the modeled device configuration
and the configuration running on the actual physical device.
                      </xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleRequestType">
      <xs:sequence>
       <xs:element name="jobRequest" type="impl:JobRequestType"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetDeltaConfigResponse">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleResponseType">
      <xs:sequence>
       <xs:element name="jobResponse" type="impl:JobResponseType"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetJobStatusRequest">
   <xs:annotation>
    <xs:documentation>Retrieves the status of the running jobs. If no job name
is specified, return all the running jobs
                    </xs:documentation>
   </xs:annotation>
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleRequestType">
      <xs:sequence>
       <xs:element name="domainId" type="xs:unsignedInt"/>
       <xs:element name="jobName" type="xs:string" minOccurs="0"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
  </xs:element>
  <xs:element name="GetJobStatusResponse">
   <xs:complexType>
    <xs:complexContent>
     <xs:extension base="core:SimpleResponseType">
      <xs:sequence>
       <xs:element name="jobStatus" type="impl:JobResponseType"
```

```
maxOccurs="unbounded"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="GetJobResultRequest">
    <xs:annotation>
     <xs:documentation>Retrieves the status of the completed jobs. If no job name
 is specified, return all the completed jobs
                    </xs:documentation>
    </xs:annotation>
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleRequestType">
       <xs:sequence>
        <xs:element name="domainId" type="xs:unsignedInt"/>
        <xs:element name="jobName" type="xs:string" minOccurs="0"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="GetJobResultResponse">
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleResponseType">
       <xs:sequence>
        <xs:element name="jobResponse" type="impl:JobResponseType"
maxOccurs="unbounded"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
  </xs:schema>
 </wsdl:types>
 <wsdl:message name="GetJobStatusRequest">
  <wsdl:part name="GetJobStatusRequest" element="impl:GetJobStatusRequest"/>
 </wsdl:message>
 <wsdl:message name="GetJobStatusResponse">
  <wsdl:part name="GetJobStatusResponse" element="impl:GetJobStatusResponse"/>
 </wsdl:message>
 <wsdl:message name="UpdateDeviceRequest">
  <wsdl:part name="UpdateDeviceRequest" element="impl:UpdateDeviceRequest"/>
 </wsdl:message>
 <wsdl:message name="UpdateDeviceResponse">
  <wsdl:part name="UpdateDeviceResponse" element="impl:UpdateDeviceResponse"/>
 </wsdl:message>
 <wsdl:message name="ImportDeviceRequest">
  <wsdl:part name="ImportDeviceRequest" element="impl:ImportDeviceRequest"/>
 </wsdl:message>
 <wsdl:message name="ImportDeviceResponse">
  <wsdl:part name="ImportDeviceResponse" element="impl:ImportDeviceResponse"/>
</wsdl:message>
<wsdl:message name="ValidateDeviceRequest">
<wsdl:part name="ValidateDeviceRequest" element="impl:ValidateDeviceRequest"/>
</wsdl:message>
<wsdl:message name="ValidateDeviceResponse">
<wsdl:part name="ValidateDeviceResponse"
element="impl:ValidateDeviceResponse"/>
```

```
 </wsdl:message>
 <wsdl:message name="GetConfigSummaryRequest">
  <wsdl:part name="GetConfigSummaryRequest"
element="impl:GetConfigSummaryRequest"/>
 </wsdl:message>
 <wsdl:message name="GetConfigSummaryResponse">
  <wsdl:part name="GetConfigSummaryResponse"
element="impl:GetConfigSummaryResponse"/>
 </wsdl:message>
 <wsdl:message name="GetRunningConfigRequest">
  <wsdl:part name="GetRunningConfigRequest"
element="impl:GetRunningConfigRequest"/>
 </wsdl:message>
 <wsdl:message name="GetRunningConfigResponse">
  <wsdl:part name="GetRunningConfigResponse"
element="impl:GetRunningConfigResponse"/>
 </wsdl:message>
 <wsdl:message name="GetDeltaConfigRequest">
  <wsdl:part name="GetDeltaConfigReqest" element="impl:GetDeltaConfigRequest"/>
 </wsdl:message>
 <wsdl:message name="GetDeltaConfigResponse">
  <wsdl:part name="GetDeltaConfigResponse" element="impl:GetDeltaConfigResponse"/>

 </wsdl:message>
 <wsdl:message name="GetJobResultRequest">
  <wsdl:part name="GetJobResultRequest" element="impl:GetJobResultRequest"/>
 </wsdl:message>
 <wsdl:message name="GetJobResultResponse">
  <wsdl:part name="GetJobResultResponse" element="impl:GetJobResultResponse"/>
 </wsdl:message>
 <wsdl:message name="CancelJobResponse">
  <wsdl:part name="CancelJobResponse" element="impl:CancelJobResponse"/>
 </wsdl:message>
 <wsdl:message name="CancelJobRequest">
  <wsdl:part name="CancelJobRequest" element="impl:CancelJobRequest"/>
 </wsdl:message>
 <wsdl:portType name="JobPortType">
  <wsdl:operation name="UpdateDeviceRequest">
   <wsdl:input message="impl:UpdateDeviceRequest"/>
   <wsdl:output message="impl:UpdateDeviceResponse"/>
  </wsdl:operation>
  <wsdl:operation name="ImportDeviceRequest">
   <wsdl:input message="impl:ImportDeviceRequest"/>
   <wsdl:output message="impl:ImportDeviceResponse"/>
  </wsdl:operation>
<wsdl:operation name="ValidateDeviceRequest">
<wsdl:input message="impl:ValidateDeviceRequest"/>
<wsdl:output message="impl:ValidateDeviceResponse"/>
</wsdl:operation>
  <wsdl:operation name="GetJobStatusRequest">
   <wsdl:input message="impl:GetJobStatusRequest"/>
   <wsdl:output message="impl:GetJobStatusResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetConfigSummaryRequest">
   <wsdl:input message="impl:GetConfigSummaryRequest"/>
   <wsdl:output message="impl:GetConfigSummaryResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetRunningConfigRequest">
   <wsdl:input message="impl:GetRunningConfigRequest"/>
   <wsdl:output message="impl:GetRunningConfigResponse"/>
  </wsdl:operation>
```

```
  <wsdl:operation name="GetDeltaConfigRequest">
   <wsdl:input message="impl:GetDeltaConfigRequest"/>
   <wsdl:output message="impl:GetDeltaConfigResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetJobResultRequest">
   <wsdl:input message="impl:GetJobResultRequest"/>
   <wsdl:output message="impl:GetJobResultResponse"/>
  </wsdl:operation>
  <wsdl:operation name="CancelJobRequest">
   <wsdl:input message="impl:CancelJobRequest"/>
   <wsdl:output message="impl:CancelJobResponse"/>
  </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="JobSoapBinding" type="impl:JobPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

  <wsdl:operation name="UpdateDeviceRequest">
   <soap:operation soapAction="urn:#UpdateDeviceRequest"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
  <wsdl:operation name="ImportDeviceRequest">
   <soap:operation soapAction="urn:#ImportDeviceRequest"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
<wsdl:operation name="ValidateDeviceRequest">
<soap:operation soapAction="urn:#ValidateDeviceRequest"/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</wsdl:operation>
  <wsdl:operation name="GetJobStatusRequest">
   <soap:operation soapAction="urn:#GetJobStatusRequest"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
  <wsdl:operation name="GetConfigSummaryRequest">
   <soap:operation soapAction="urn:#GetConfigSummaryRequest"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
```

```
  <wsdl:operation name="GetRunningConfigRequest">
   <soap:operation soapAction="urn:#GetRunningConfigRequest"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
  <wsdl:operation name="GetDeltaConfigRequest">
   <soap:operation soapAction="urn:#GetDeltaConfigRequest"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
  <wsdl:operation name="GetJobResultRequest">
   <soap:operation soapAction="urn:#GetJobResultRequest"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
  <wsdl:operation name="CancelJobRequest">
   <soap:operation soapAction="urn:#CancelJobRequest"/>
   <wsdl:input>
    <soap:body use="literal"/>
   </wsdl:input>
   <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="JobService">
  <wsdl:port name="Job" binding="impl:JobSoapBinding">
   <soap:address location="csp://webproxy/nsm/service/JobService"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

CHAPTER 16

System Service API WSDL

This chapter describes the System Service API WSDL.

- WSDL File on page 173

WSDL File

The WSDL definition file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:impl="http://juniper.net/webproxy/systemservice"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:core="http://juniper.net/core"
 xmlns:ns="http://schemas.xmlsoap.org/soap/encoding/" name="SystemService"
targetNamespace="http://juniper.net/webproxy/systemservice">
 <wsdl:types>
  <xs:schema version="1.0" elementFormDefault="qualified"
targetNamespace="http://juniper.net/webproxy/systemservice"
xmlns="http://juniper.net/webproxy/systemservice"
xmlns:core="http://juniper.net/core"
xmlns:impl="http://juniper.net/webproxy/systemservice">
   <xs:import namespace="http://juniper.net/core"
schemaLocation="common/BaseMessages.xsd"/>
   <xs:simpleType name="LoginStatusCodeType">
    <xs:restriction base="xs:token">
     <xs:enumeration value="Success"/>
     <xs:enumeration value="Failure"/>
     <xs:enumeration value="Challenge"/>
    </xs:restriction>
   </xs:simpleType>
   <xs:complexType name="LoginStatus">
    <xs:sequence>
     <xs:element name="status" type="impl:LoginStatusCodeType"/>
     <xs:element name="challenge" type="xs:string" minOccurs="0"/>
    </xs:sequence>
   </xs:complexType>
   <xs:element name="LoginRequest">
    <xs:annotation>
     <xs:documentation>Login into the system

domainName: the domain to login
userName: the user name
password: the password
```

```
                        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleRequestType">
       <xs:sequence>
        <xs:element name="domainName" type="xs:string"/>
        <xs:element name="userName" type="xs:string"/>
        <xs:element name="password" type="xs:string"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:complexType name="LoginResponseType">
    <xs:annotation>
     <xs:documentation>The response of the login request

loginStatus: if the status is Success, a valid token is returned for requests
followed, if the status is Challenge, the response to the challenge shall be sent
authToken: the token used by the further requests
                        </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
     <xs:extension base="core:SimpleResponseType">
      <xs:sequence>
       <xs:element name="loginStatus" type="impl:LoginStatus"/>
       <xs:element name="authToken" type="core:AuthTokenType" minOccurs="0"/>
      </xs:sequence>
     </xs:extension>
    </xs:complexContent>
   </xs:complexType>
   <xs:element name="LoginResponse" type="impl:LoginResponseType"/>
   <xs:element name="RespondToChallengeRequest">
    <xs:annotation>
     <xs:documentation>Send the response to the challenge
                        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleRequestType">
       <xs:sequence>
        <xs:element name="challengeResponse" type="xs:string"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="RespondToChallengeResponse" type="impl:LoginResponseType"/>

   <xs:element name="LogoutRequest">
    <xs:complexType>
     <xs:complexContent>
      <xs:extension base="core:SimpleRequestType"/>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:complexType name="ServiceDescType">
    <xs:sequence>
     <xs:element name="name" type="xs:string"/>
     <xs:element name="version" type="xs:string"/>
```

```
     <xs:element name="definition" type="xs:string"/>
    </xs:sequence>
   </xs:complexType>
   <xs:element name="GetSystemInfoRequest">
    <xs:complexType>
     <xs:annotation>
      <xs:documentation>Get the system informations: the service description and
 all the accessible domain ids and names. If no service name is specified, return
 the service description of all services
                        </xs:documentation>
     </xs:annotation>
     <xs:complexContent>
      <xs:extension base="core:SimpleRequestType">
       <xs:sequence>
        <xs:element name="serviceName" type="xs:string" minOccurs="0"
maxOccurs="unbounded"/>
       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
   <xs:element name="GetSystemInfoResponse">
    <xs:complexType>
     <xs:annotation>
      <xs:documentation>Return the service list and all the accessible domain ids
 and names
                    </xs:documentation>
     </xs:annotation>
     <xs:complexContent>
      <xs:extension base="core:SimpleResponseType">
       <xs:sequence>
        <xs:element name="serviceDesc" type="impl:ServiceDescType" minOccurs="0"
 maxOccurs="unbounded"/>
        <xs:element name="domainName" type="xs:string" maxOccurs="unbounded"/>
       <xs:element name="domainId" type="xs:unsignedInt" maxOccurs="unbounded"/>

       </xs:sequence>
      </xs:extension>
     </xs:complexContent>
    </xs:complexType>
   </xs:element>
  </xs:schema>
 </wsdl:types>
 <wsdl:message name="LoginRequest">
  <wsdl:part name="LoginRequest" element="impl:LoginRequest"/>
 </wsdl:message>
 <wsdl:message name="LoginResponse">
  <wsdl:part name="LoginRequest" element="impl:LoginResponse"/>
 </wsdl:message>
 <wsdl:message name="RespondToChallengeRequest">
  <wsdl:part name="RespondToChallengeRequest"
element="impl:RespondToChallengeRequest"/>
 </wsdl:message>
 <wsdl:message name="RespondToChallengeResponse">
  <wsdl:part name="RespondToChallengeResponse"
element="impl:RespondToChallengeResponse"/>
 </wsdl:message>
 <wsdl:message name="LogoutRequest">
  <wsdl:part name="LogoutRequest" element="impl:LogoutRequest"/>
 </wsdl:message>
 <wsdl:message name="GetSystemInfoRequest">
```

```
 <wsdl:part name="GetSystemInfoRequest" element="impl:GetSystemInfoRequest"/>
</wsdl:message>
<wsdl:message name="GetSystemInfoResponse">
 <wsdl:part name="GetSystemInfoResponse" element="impl:GetSystemInfoResponse"/>

</wsdl:message>
<wsdl:portType name="SystemPortType">
 <wsdl:operation name="LoginRequest">
  <wsdl:input message="impl:LoginRequest"/>
  <wsdl:output message="impl:LoginResponse"/>
 </wsdl:operation>
 <wsdl:operation name="RespondToChallengeRequest">
  <wsdl:input message="impl:RespondToChallengeRequest"/>
  <wsdl:output message="impl:RespondToChallengeResponse"/>
 </wsdl:operation>
 <wsdl:operation name="LogoutRequest">
  <wsdl:input message="impl:LogoutRequest"/>
 </wsdl:operation>
 <wsdl:operation name="GetSystemInfoRequest">
  <wsdl:input message="impl:GetSystemInfoRequest"/>
  <wsdl:output message="impl:GetSystemInfoResponse"/>
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="SystemSoapBinding" type="impl:SystemPortType">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

 <wsdl:operation name="LoginRequest">
  <soap:operation soapAction="urn:#LoginRequest"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="RespondToChallengeRequest">
  <soap:operation soapAction="urn:#RespondToChallengeRequest"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="LogoutRequest">
  <soap:operation soapAction="urn:#LogoutRequest"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
 </wsdl:operation>
 <wsdl:operation name="GetSystemInfoRequest">
  <soap:operation soapAction="urn:#GetSystemInfoRequest"/>
  <input>
   <soap:body use="literal"/>
  </input>
  <output>
   <soap:body use="literal"/>
  </output>
 </wsdl:operation>
</wsdl:binding>
<wsdl:service name="SystemService">
```

```
  <wsdl:port name="System" binding="impl:SystemSoapBinding">
   <soap:address location="http://localhost:8080/axis2/services/SystemService"/>

  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

CHAPTER 17

Data Centric API WSDL

This chapter describes the Data Centric API WSDL.

- WSDL File on page 179

WSDL File

The WSDL definition file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2007 rel. 3 sp1 (http://www.altova.com) by Shaogang Chen
 (Juniper Networks, Inc.) -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:impl="http://juniper.net/nbiservice/datacentricservice"
xmlns:ns="http://juniper.net/webproxy/datacentricservice"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:core="http://juniper.net/core"
 xmlns:ns1="http://schemas.xmlsoap.org/soap/encoding/" name="DataCentricService"
 targetNamespace="http://juniper.net/webproxy/datacentricservice">
 <wsdl:types>
  <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://juniper.net/nbiservice/datacentricservice"
elementFormDefault="qualified" version="1.0">
   <import namespace="http://juniper.net/core"
schemaLocation="common/BaseMessages.xsd"/>
   <simpleType name="BuiltInViewType">
    <restriction base="xs:string">
     <enumeration value="DefaultView"/>
                  <enumeration value="XDBView"/>
               </restriction>
   </simpleType>
   <complexType name="NameValueType">
    <sequence>
     <element name="name" type="xs:string"/>
     <element name="value" type="xs:string"/>
    </sequence>
   </complexType>
   <complexType name="ViewFilterType">
    <choice>
     <element name="metadataOnly" type="xs:boolean"/>
     <element name="filter" type="core:OpaqueDataType"/>
    </choice>
   </complexType>
   <complexType name="ObjectViewFilterType">
```

```
    <sequence>
     <element name="category" type="xs:string"/>
     <element name="viewFilter" type="impl:ViewFilterType"/>
    </sequence>
   </complexType>
   <complexType name="InsertObjectViewType">
    <sequence>
     <element name="objecData" type="core:ObjectDataType"/>
     <element name="category" type="xs:string"/>
     <element name="domainId" type="xs:unsignedShort"/>
     <element name="objectIdentifier" type="xs:unsignedInt" minOccurs="0"/>
    </sequence>
   </complexType>
   <complexType name="RenameObjectViewType">
    <sequence>
     <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
     <element name="newName" type="xs:string"/>
    </sequence>
   </complexType>
   <complexType name="ReplaceObjectViewType">
    <sequence>
     <element name="objecData" type="core:ObjectDataType"/>
     <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
    </sequence>
   </complexType>
<complexType name="DeleteMgmtIntViewType">
        <sequence>
        <element name="objecData" type="core:ObjectDataType"/>
        <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
       <xs:element name="nodeName" type="xs:string" />
       <xs:element name="nodetype" type="xs:string" />
       <xs:element name="groupName" type="xs:string" />
      </sequence>
    </complexType>
   <complexType name="DeleteObjectViewType">
    <sequence>
     <element name="objectIdentifier" type="core:ObjectIdentifierType"
maxOccurs="unbounded"/>
    </sequence>
   </complexType>
   <complexType name="UpdateObjectViewType">
    <sequence>
     <element name="objectIdentifier" type="core:ObjectIdentifierType"
maxOccurs="unbounded"/>
     <element name="objectModification" type="core:ObjectModificationType"/>
    </sequence>
   </complexType>


<complexType name="ReplaceDeviceSubNodeViewType">
   <sequence>
                <element name="objecData" type="core:ObjectDataType"/>
                 <element name="objectIdentifier"
type="core:ObjectIdentifierType"/>
  <xs:element name="nodeName" type="xs:string" />
  <xs:element name="nodetype" type="xs:string" />
   </sequence>
   </complexType>
    <complexType name="AppendDeviceSubNodeViewType">
    <sequence>
               <element name="objecData" type="core:ObjectDataType"/>
```

```
            <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
   <xs:element name="nodetype" type="xs:string" />
      </sequence>
      </complexType>
      <complexType name="DeleteRuleViewType">
      <sequence>
                  <element name="objecData" type="core:ObjectDataType"/>
   <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
  <xs:element name="preferredId" type="xs:string" minOccurs="0"/>
                  <xs:element name="policyName" type="xs:string" minOccurs="0"/>

                                                          </sequence>
      </complexType>
        <complexType name="ReplaceNATSubNodeViewType">
        <sequence>
  <element name="objecData" type="core:ObjectDataType"/>
  <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
  <element name="natobject" type="core:NATNodeType"/>
  <element name="natnodename" type="xs:string"/>
         </sequence>
       </complexType>
      <complexType name="AppendNATSubNodeViewType">
                                                          <sequence>
                                                                <element
name="objecData" type="core:ObjectDataType"/>
  <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
  <element name="natobject" type="core:NATNodeType"/>
     </sequence>
 </complexType>
 <complexType name="AppendRouteSubNodeViewType">
 <sequence>
  <element name="objecData" type="core:ObjectDataType"/>
  <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
  <element name="vroutername" type="xs:string"/>
 </sequence>
 </complexType>
 <complexType name="ReplaceRouteSubNodeViewType">
 <sequence>
  <element name="objecData" type="core:ObjectDataType"/>
  <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
  <element name="vroutername" type="xs:string"/>
  <element name="routename" type="xs:string"/>
 </sequence>
 </complexType>
   <complexType name="ModifyViewCommandType">
    <choice>
     <element name="insertObject" type="impl:InsertObjectViewType"/>
     <element name="renameObject" type="impl:RenameObjectViewType"/>
     <element name="replaceObject" type="impl:ReplaceObjectViewType"/>
     <element name="deleteObject" type="impl:DeleteObjectViewType"/>
     <element name="updateObject" type="impl:UpdateObjectViewType"/>
 <element name="replaceDeviceSubNodeObject"
type="impl:ReplaceDeviceSubNodeViewType"/>
 <element name="appendDeviceSubNodeObject"
type="impl:AppendDeviceSubNodeViewType"/>
 <element name="deleteRuleObject" type="impl:DeleteRuleViewType"/>
 <element name="replaceNATSubNodeObject" type="impl:ReplaceNATSubNodeViewType"/>

 <element name="appendNATSubNodeObject" type="impl:AppendNATSubNodeViewType"/>
 <element name="appendRouteSubNodeObject" type="impl:AppendRouteSubNodeViewType"/>
```

```
 <element name="replaceRouteSubNodeObject"
type="impl:ReplaceRouteSubNodeViewType"/>
  <element name="deleteMgmtIntObject" type="impl:DeleteMgmtIntViewType"/>
  </choice>
   </complexType>
<element name="GetObjectViewByCategoryRequest">
    <annotation>
     <documentation>Get the objects in one category

category: the schema name of the object
domainId: the domain of the schema
dbVersionId: the version of the data respository
objectFilter: the filter to be applied on the result
view: the transformation of the object, the default view means that the object
returned follows the schema exactly, and no transformation is applied on the
object
property: the parameters of the transformation
                        </documentation>
    </annotation>
    <complexType>
     <complexContent>
      <extension base="core:SimpleRequestType">
       <sequence>
        <element name="category" type="xs:string"/>
        <element name="domainId" type="xs:unsignedShort" minOccurs="0"/>
                  <element name="rtveDefaults" type="xs:boolean" minOccurs="0"/>

        <element name="dbVersionId" type="xs:unsignedInt" minOccurs="0"/>
        <element name="objectFilter" type="impl:ViewFilterType" minOccurs="0"/>
        <element name="view" type="xs:string" minOccurs="0"/>
        <element name="property" type="impl:NameValueType" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="GetObjectViewByCategoryResponse">
    <complexType>
     <complexContent>
      <extension base="core:SimpleResponseType">
       <sequence>
        <element name="object" type="core:ObjectType" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="GetObjectViewByIdRequest">
    <annotation>
     <documentation>Get the objects based on the id

objectIdentifier: the id of the object to be retrieved
dbVersionId: the version of the data respository
objectFilter: the filter to be applied on the result
view: the transformation of the object, the default view means that the object
returned follows the schema exactly, and no transformation is applied on the
object
property: the parameters of the transformation
                        </documentation>
```

```
    </annotation>
    <complexType>
     <complexContent>
      <extension base="core:SimpleRequestType">
       <sequence>
        <element name="objectIdentifier" type="core:ObjectIdentifierType"
maxOccurs="unbounded"/>
        <element name="dbVersionId" type="xs:unsignedInt" minOccurs="0"/>
       <element name="objectFilter" type="impl:ObjectViewFilterType" minOccurs="0"
 maxOccurs="unbounded"/>
                 <element name="rtveDefaults" type="xs:boolean" minOccurs="0"/>
        <element name="view" type="xs:string" minOccurs="0"/>
        <element name="property" type="impl:NameValueType" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="GetObjectViewByIdResponse">
    <complexType>
     <complexContent>
      <extension base="core:SimpleResponseType">
       <sequence>
        <element name="object" type="core:ObjectType" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="ResolveObjectReferenceRequest">
    <annotation>
     <documentation>Resolve the object reference.

objectReference: the object reference to be resolved
dbVersionId: the version of the data respository
objectFilter: the filter to be applied on the result
                        </documentation>
    </annotation>
    <complexType>
     <complexContent>
      <extension base="core:SimpleRequestType">
       <sequence>
        <element name="objectReference" type="xs:string" maxOccurs="unbounded"/>

        <element name="dbVersionId" type="xs:unsignedInt" minOccurs="0"/>
        <element name="objectFilter" type="impl:ObjectViewFilterType"
minOccurs="0"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="ResolveObjectReferenceResponse">
    <complexType>
     <complexContent>
      <extension base="core:SimpleResponseType">
       <sequence>
        <element name="object" type="core:ObjectType" maxOccurs="unbounded"/>
       </sequence>
```

```
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="GetObjectDependentRequest">
    <annotation>
     <documentation>Get the objects that refer to the object specified in the
request.

objectIdentifier: the id of the object
dbVersionId: the version of the data respository
metadataOnly: if true, only metadata is returned. Otherwise, the whole object is
 returned
     </documentation>
    </annotation>
    <complexType>
     <complexContent>
      <extension base="core:SimpleRequestType">
       <sequence>
        <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
        <element name="dbVersionId" type="xs:unsignedInt" minOccurs="0"/>
        <element name="metadataOnly" type="xs:boolean" minOccurs="0"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="GetObjectDependentResponse">
    <complexType>
     <complexContent>
      <extension base="core:SimpleResponseType">
       <sequence>
        <element name="object" type="core:ObjectType" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="QueryObjectViewRequest">
    <annotation>
     <documentation>Query the objects. The query expression is specified by
simpleQuery.

simpleQuery: the query expression
dbVersionId: the version of the data respository
objectFilter: the filter to be applied on the result
view: the transformation of the object, the default view means that the object
returned follows the schema exactly, and no transformation is applied on the
object
property: the parameters of the transformation
                     </documentation>
    </annotation>
    <complexType>
     <complexContent>
      <extension base="core:SimpleRequestType">
       <sequence>
        <element name="simpleQuery" type="core:SimpleQueryType"/>
        <element name="dbVersionId" type="xs:unsignedInt" minOccurs="0"/>
        <element name="objectFilter" type="impl:ObjectViewFilterType"
minOccurs="0"/>
```

```
        <element name="view" type="xs:string" minOccurs="0"/>
        <element name="property" type="impl:NameValueType" minOccurs="0"
maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="QueryObjectViewResponse">
    <annotation>
     <documentation>The response of the query
     </documentation>
    </annotation>
    <complexType>
     <complexContent>
      <extension base="core:SimpleResponseType">
       <sequence>
        <element name="resultSet" type="core:ResultSetType"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="ModifyObjectViewRequest">
    <annotation>
     <documentation>Object modification. All the commands in the request are
executed in one transaction
     </documentation>
    </annotation>
    <complexType>
     <complexContent>
      <extension base="core:SimpleRequestType">
       <sequence>
                                <element name="view" type="xs:string"
minOccurs="0"/>
        <element name="property" type="impl:NameValueType" minOccurs="0"
maxOccurs="unbounded"/>
                                <element name="command"
type="impl:ModifyViewCommandType" maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="ModifyObjectViewResponse">
    <complexType>
     <complexContent>
      <extension base="core:SimpleResponseType">
       <sequence>
        <element name="modification" type="core:ModificationResponseType"
minOccurs="0" maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="LockObjectViewRequest">
    <complexType>
     <complexContent>
      <extension base="core:SimpleRequestType">
       <sequence>
```

```
        <element name="objectIdentifier" type="core:ObjectIdentifierType"
maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <element name="UnlockObjectViewRequest">
    <complexType>
     <complexContent>
      <extension base="core:SimpleRequestType">
       <sequence>
        <element name="objectIdentifier" type="core:ObjectIdentifierType"
maxOccurs="unbounded"/>
       </sequence>
      </extension>
     </complexContent>
    </complexType>
   </element>
   <complexType name="LockingViewResponseType">
    <complexContent>
     <extension base="core:SimpleResponseType">
      <sequence>
       <element name="objectLockStatus" type="core:LockStatusType"
maxOccurs="unbounded"/>
      </sequence>
     </extension>
    </complexContent>
   </complexType>
   <element name="LockObjectViewResponse" type="impl:LockingViewResponseType"/>
   <element name="UnlockObjectViewResponse" type="impl:LockingViewResponseType"/>

  </schema>
 </wsdl:types>
 <wsdl:message name="GetObjectViewByIdRequest">
  <wsdl:part name="GetObjecViewByIdRequest"
element="impl:GetObjectViewByIdRequest"/>
 </wsdl:message>
 <wsdl:message name="GetObjectViewByIdResponse">
  <wsdl:part name="GetObjectViewByIdResponse"
element="impl:GetObjectViewByIdResponse"/>
 </wsdl:message>
 <wsdl:message name="GetObjectViewByCategoryRequest">
  <wsdl:part name="GetObjectViewByCategoryRequest"
element="impl:GetObjectViewByCategoryRequest"/>
 </wsdl:message>
 <wsdl:message name="GetObjectViewByCategoryResponse">
  <wsdl:part name="GetObjectViewByCategoryResponse"
element="impl:GetObjectViewByCategoryResponse"/>
 </wsdl:message>
 <wsdl:message name="ModifyObjectViewRequest">
  <wsdl:part name="ModifyObjectViewRequest"
element="impl:ModifyObjectViewRequest"/>
 </wsdl:message>
 <wsdl:message name="ModifyObjectViewResponse">
  <wsdl:part name="ModifyObjectViewResponse"
element="impl:ModifyObjectViewResponse"/>
 </wsdl:message>
 <wsdl:message name="ResolveObjectReferenceRequest">
  <wsdl:part name="ResolveObjectReferenceRequest"
element="impl:ResolveObjectReferenceRequest"/>
```

```
 </wsdl:message>
 <wsdl:message name="ResolveObjectReferenceResponse">
  <wsdl:part name="ResolveObjectReferenceResponse"
element="impl:ResolveObjectReferenceResponse"/>
 </wsdl:message>
 <wsdl:message name="QueryObjectViewRequest">
  <wsdl:part name="QueryObjectViewRequest" element="impl:QueryObjectViewRequest"/>

 </wsdl:message>
 <wsdl:message name="QueryObjectViewResponse">
  <wsdl:part name="QueryObjectViewResponse"
element="impl:QueryObjectViewResponse"/>
 </wsdl:message>
 <wsdl:message name="GetObjectDependentRequest">
  <wsdl:part name="GetObjectDependentRequest"
element="impl:GetObjectDependentRequest"/>
 </wsdl:message>
 <wsdl:message name="GetObjectDependentResponse">
  <wsdl:part name="GetObjectDependentResponse"
element="impl:GetObjectDependentResponse"/>
 </wsdl:message>
 <wsdl:message name="LockObjectViewRequest">
  <wsdl:part name="LockObjectViewRequest" element="impl:LockObjectViewRequest"/>

 </wsdl:message>
 <wsdl:message name="LockObjectViewResponse">
  <wsdl:part name="LockObjectViewResponse" element="impl:LockObjectViewResponse"/>

 </wsdl:message>
 <wsdl:message name="UnlockObjectViewRequest">
  <wsdl:part name="UnlockObjectViewRequest"
element="impl:UnlockObjectViewRequest"/>
 </wsdl:message>
 <wsdl:message name="UnlockObjectViewResponse">
  <wsdl:part name="UnlockObjectViewResponse"
element="impl:UnlockObjectViewResponse"/>
 </wsdl:message>
 <wsdl:portType name="DataCentricPortType">
  <wsdl:operation name="GetObjectViewByIdRequest">
   <wsdl:input message="ns:GetObjectViewByIdRequest"/>
   <wsdl:output message="ns:GetObjectViewByIdResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetObjectViewByCategoryRequest">
   <wsdl:input message="ns:GetObjectViewByCategoryRequest"/>
   <wsdl:output message="ns:GetObjectViewByCategoryResponse"/>
  </wsdl:operation>
  <wsdl:operation name="ModifyObjectViewRequest">
   <wsdl:input message="ns:ModifyObjectViewRequest"/>
   <wsdl:output message="ns:ModifyObjectViewResponse"/>
  </wsdl:operation>
  <wsdl:operation name="ResolveObjectReferenceRequest">
   <wsdl:input message="ns:ResolveObjectReferenceRequest"/>
   <wsdl:output message="ns:ResolveObjectReferenceResponse"/>
  </wsdl:operation>
  <wsdl:operation name="QueryObjectViewRequest">
   <wsdl:input message="ns:QueryObjectViewRequest"/>
   <wsdl:output message="ns:QueryObjectViewResponse"/>
  </wsdl:operation>
  <wsdl:operation name="GetObjectDependentRequest">
   <wsdl:input message="ns:GetObjectDependentRequest"/>
   <wsdl:output message="ns:GetObjectDependentResponse"/>
```

```
 </wsdl:operation>
 <wsdl:operation name="LockObjectViewRequest">
  <wsdl:input message="ns:LockObjectViewRequest"/>
  <wsdl:output message="ns:LockObjectViewResponse"/>
 </wsdl:operation>
 <wsdl:operation name="UnlockObjectViewRequest">
  <wsdl:input message="ns:UnlockObjectViewRequest"/>
  <wsdl:output message="ns:UnlockObjectViewResponse"/>
 </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="DataCentricSoapBinding" type="ns:DataCentricPortType">
 <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

 <wsdl:operation name="GetObjectViewByIdRequest">
  <soap:operation soapAction="urn:#GetObjectViewByIdRequest"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="GetObjectViewByCategoryRequest">
  <soap:operation soapAction="urn:#GetObjectViewByCategoryRequest"/>
  <wsdl:input>
   <soap:body use="literal"/>
  </wsdl:input>
  <wsdl:output>
   <soap:body use="literal"/>
  </wsdl:output>
 </wsdl:operation>
 <wsdl:operation name="ModifyObjectViewRequest">
  <soap:operation soapAction="urn:#ModifyObjectViewRequest"/>
  <input>
   <soap:body use="literal"/>
  </input>
  <outpout>
   <soap:body use="literal"/>
  </output>
 </wsdl:operation>
 <wsdl:operation name="ResolveObjectReferenceRequest">
  <soap:operation soapAction="urn:#ResolveObjectReferenceRequest"/>
  <input>
   <soap:body use="literal"/>
  </input>
  <outpout>
   <soap:body use="literal"/>
  </output>
 </wsdl:operation>
 <wsdl:operation name="QueryObjectViewRequest">
  <soap:operation soapAction="urn:#QueryObjectViewRequest"/>
  <input>
   <soap:body use="literal"/>
  </input>
  <outpout>
   <soap:body use="literal"/>
  </output>
 </wsdl:operation>
 <wsdl:operation name="GetObjectDependentRequest">
  <soap:operation soapAction="urn:#GetObjectDependentRequest"/>
  <input>
```

```
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
  <wsdl:operation name="LockObjectViewRequest">
   <soap:operation soapAction="urn:#LockObjectViewRequest"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
  <wsdl:operation name="UnlockObjectViewRequest">
   <soap:operation soapAction="urn:#UnLockObjectViewRequest"/>
   <input>
    <soap:body use="literal"/>
   </input>
   <output>
    <soap:body use="literal"/>
   </output>
  </wsdl:operation>
 </wsdl:binding>
 <wsdl:service name="DataCentricService">
  <documentation> DataCentric service provides the API for data transformation.
The schemas of the data to be transformed are specified in the seperate
documentation.
</documentation>
  <wsdl:port name="DataCentric" binding="ns:DataCentricSoapBinding">
   <soap:address location="csp://nbiservice/nsm/service/DataCentricService"/>
  </wsdl:port>
 </wsdl:service>
</wsdl:definitions>
```

CHAPTER 18

Log Service API WSDL

This chapter describes the Log Service API WSDL.

- WSDL File on page 191

WSDL File

The WSDL definition file is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 rel. 3 sp1 (http://www.altova.com) by aao (Juniper
 Networks, Inc.) -->
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:impl="http://juniper.net/nbiservice/logservice"
xmlns:ns="http://juniper.net/webproxy/logservice"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:core="http://juniper.net/core"
 targetNamespace="http://juniper.net/webproxy/logservice" name="LogService">
    <wsdl:types>
        <xs:schema xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
targetNamespace="http://juniper.net/nbiservice/logservice">
            <import namespace="http://juniper.net/core"
schemaLocation="common/BaseMessages.xsd"/>
            <element name="GetPacketDataRequest">
                <complexType>
                    <complexContent>
                        <extension base="core:SimpleRequestType">
                            <sequence>
                                <element name="dayId" type="xs:unsignedInt"/>
                               <element name="recordNum" type="xs:unsignedInt"/>

                            </sequence>
                        </extension>
                    </complexContent>
                </complexType>
            </element>
            <element name="GetPacketDataResponse">
                <complexType>
                    <complexContent>
                        <extension base="core:SimpleResponseType">
                            <sequence>
                                <element name="numPackets" type="xs:unsignedInt"
 minOccurs="0"/>
```

```
                                <element name="triggerPacket" type="xs:unsignedInt"
 minOccurs="0"/>
                                   <element name="data" type="xs:string"
minOccurs="0"/>
                               </sequence>
                           </extension>
                       </complexContent>
                   </complexType>
               </element>
           </xs:schema>
       </wsdl:types>
       <wsdl:message name="GetPacketDataRequest">
           <wsdl:part name="GetPacketDataRequest"
element="impl:GetPacketDataRequest"/>
       </wsdl:message>
       <wsdl:message name="GetPacketDataResponse">
           <wsdl:part name="GetPacketDataResponse"
element="impl:GetPacketDataResponse"/>
       </wsdl:message>
       <wsdl:portType name="LogPortType">
           <wsdl:operation name="GetPacketDataRequest">
               <wsdl:input message="ns:GetPacketDataRequest"/>
               <wsdl:output message="ns:GetPacketDataResponse"/>
           </wsdl:operation>
       </wsdl:portType>
       <wsdl:binding name="LogSoapBinding" type="ns:LogPortType">
           <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
           <wsdl:operation name="GetPacketDataRequest">
               <soap:operation soapAction="urn:#GetPacketDataRequest"/>
               <wsdl:input>
                   <soap:body use="literal"/>
               </wsdl:input>
               <wsdl:output>
                   <soap:body use="literal"/>
               </wsdl:output>
           </wsdl:operation>
       </wsdl:binding>
       <wsdl:service name="LogService">
           <wsdl:port name="Log" binding="ns:LogSoapBinding">
               <soap:address location="csp://nbiservice/nsm/service/LogService"/>
           </wsdl:port>
       </wsdl:service>
</wsdl:definitions>
```

PART 6

Index

- Index on page 195

Index

A

API
- Data Centric Service....8
- Job Service....11
- Log Service....13
- operations....7
- System Service....7

API data objects....17
attack
- API logs....28
- backdoor....25
- critical....28
- IDP will exempt....29
- information....28
- objects....40
- service used in....27
- severity of IDP rulebase....29
- tag....28

authentication....4, 99
authorization....4

B

backdoor_type
- action....28
- comments....27
- customOptions_collection....27
- dst_addr_collection....27
- dst_addr_negate....27
- dst_zone_collection....27
- enabled....27
- log....28
- log-actions....28
- op....28
- preferred-id....27
- rb-link....27
- ruleno....27
- service....27
- seslog....29
- severity....29
- src_addr_collection....27
- src_addr_negate....27
- src_zone_collection....27
- target_collection....29
- vlan....28

C

CommonDataTypes.xsd....4
customer support....xiii
- contacting JTAC....xiii

D

Data Centric API....101
- GetObjectDependentRequest....9
- GetObjectViewByCategoryRequest....9
- GetObjectViewByIdRequest....10
- LockObjectRequest....10
- ModifyObjectViewRequest....10
- QueryObjectViewRequest....11
- ResolveObjectReferenceRequest....11
- UnlockObjectRequest....10
- using for device management....8, 125
- using for policy management....8
- using for shared object management....8
- using to manage shared objects....111

data type
- AuthTokenType....19
- ConversationContextType....20
- DataFormatType....18
- DomainIdOrNameType....19
- ErrorType....4
- ObjectDataType....19
- ObjectIdentifierType....18
- ObjectIdOrNameType....19
- ObjectMetadataType....19
- ObjectType....19
- OpaqueDataType....18
- ProgressType....20
- SequenceType....19
- StatusCodeType....19
- SubObjectDataType....19

device configuration....xi, 11

E

error handling....4

F

firewall policy type
- action....33
- anti-spam....37
- application....36

application-services....37
attack-policy....36
auths....35
av....37
count....34
grp....37
ha_session_backup....35
idp....36
log....34
log-actions....34
name_....33
nat....34
nat-dip....34
no-idle-reset....35
pol_name....33
preferred-id....34
schedule....35
service_collection....33
target_collection....34
traffic....35
url-blk....34
url-protocol....34

I

IDP
attacks exempted....31
detects interactive traffic....25
rulebase....25, 29, 47
rules....37
IDP Sensor....46
See also IDP
idppolicy_type
action....39
attacks....40
comments....39
customOptions_collection....39
diffserv....39
dst_addr_collection....39
dst_addr_negate....39
dst_zone_collection....39
enabled....39
ipaction....40
log....40
log-actions....40
preferred-id....39
rb-link....39
ruleno....39
service....39
seslog....40
severity....40
src_addr_collection....39
src_addr_negate....39
src_zone_collection....39
target_collection....40
terminal....39
vlan....40
Intrusion Detection and Prevention (IDP)
supported on devices *See* IDP

J

Job Service API
GetConfigSummaryRequest....12
GetDeltaConfigRequest....13
GetJobResultRequest....13
GetJobStatusRequest....13
GetRunningConfigRequest....12
ImportDeviceRequest....12
UpdateDeviceRequest....11
using for job management....11, 117

L

log
action settings....47
alert....40
attack....44
audit....20
data and packet data....14
event....14
GTP....46
packet....40
packet data retrieval....3
packets....29
record....29
Log Service API
GetPacketDataRequest....14
Log Viewer
using to view logs *See* log

M

message types
SimpleRequest....21
SimpleResponse....21

N

network address translation (NAT)....34
NSM Security
data model....23
rulebases....25

NSM server
catches application-level errors....4
connecting to....4
login....7
logout....8
nsmpolicy_collection_type
policy_type....24
See also policy_type

P

policy_type
accesstype....24
comment....24
createFrom....24
name_....24
rulebases....24
port scan
distributed....46
TCP....46
UDP....46

R

rb_backdoor_type
name_....27
rules....27
rules_collection....27
rb_idp_type
name_....38
next_preferred_id....38
rowcountperrule_collection....38
rules_collection....38
rule
attach matching....28
backdoor....26
global....25
row count per....38
rule ID....27, 31, 34, 39
service object....27
rulebases
backdoor....25
exempt....29
firewall....31
IDP....37
multicast....40
rules....25
SYN protector....42

S

security policy
rulebases....25
SOAP/HTTPS....3
subtree filter
attribute matching expressions....9
containment nodes....9
content matching nodes....9
namespace selection....9
selection nodes....9
support, technical *See* technical support
System Service API....99
GetSystemInfoRequest....8
LoginRequest....7
LogoutRequest....8
RespondToChallengeRequest....8
using for authentication....7
using for authorization....7

T

technical support
contacting JTAC....xiii
third-party developers....3
traffic
flow....25
matching to rules....25

V

vlan
messages in specified....28
tag....47
VLAN tag....31

W

Web Service Definition Language....3
WSDL....3
Data Centric API....179
Job Service API....163
Log Service API....191
System Service API....173

X

XML
schema....3
subtree filter....9
See also subtree filter
XSD....3
definition files....23