



# Juniper Networks Network and Security Manager

## API Guide

Release  
2010.1



Published: 2011-04-01  
Revision

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

This product includes the Envoy SNMP Engine, developed by Epilogue Technology, an Integrated Systems Company. Copyright © 1986–1997, Epilogue Technology Corporation. All rights reserved. This program and its documentation were developed at private expense, and no part of them is in the public domain.

This product includes memory allocation software developed by Mark Moraes, copyright © 1988, 1989, 1993, University of Toronto.

This product includes FreeBSD software developed by the University of California, Berkeley, and its contributors. All of the documentation and software included in the 4.4BSD and 4.4BSD-Lite Releases is copyrighted by the Regents of the University of California. Copyright © 1979, 1980, 1983, 1986, 1988, 1989, 1991, 1992, 1993, 1994. The Regents of the University of California. All rights reserved.

GateD software copyright © 1995, the Regents of the University. All rights reserved. Gate Daemon was originated and developed through release 3.0 by Cornell University and its collaborators. Gated is based on Kirton's EGP, UC Berkeley's routing daemon (routed), and DCN's HELLO routing protocol. Development of Gated has been supported in part by the National Science Foundation. Portions of the GateD software copyright © 1988, Regents of the University of California. All rights reserved. Portions of the GateD software copyright © 1991, D. L. S. Associates.

This product includes software developed by Maker Communications, Inc., copyright © 1996, 1997, Maker Communications, Inc.

Juniper Networks, Junos, Steel-Belted Radius, NetScreen, and ScreenOS are registered trademarks of Juniper Networks, Inc. in the United States and other countries. The Juniper Networks Logo, the Junos logo, and JunosE are trademarks of Juniper Networks, Inc. All other trademarks, service marks, registered trademarks, or registered service marks are the property of their respective owners.

Juniper Networks assumes no responsibility for any inaccuracies in this document. Juniper Networks reserves the right to change, modify, transfer, or otherwise revise this publication without notice.

Products made or sold by Juniper Networks or components thereof might be covered by one or more of the following patents that are owned by or licensed to Juniper Networks: U.S. Patent Nos. 5,473,599, 5,905,725, 5,909,440, 6,192,051, 6,333,650, 6,359,479, 6,406,312, 6,429,706, 6,459,579, 6,493,347, 6,538,518, 6,538,899, 6,552,918, 6,567,902, 6,578,186, and 6,590,785.

*Network and Security Manager API Guide*

Copyright © 2011, Juniper Networks, Inc.

All rights reserved.

Writing: Jim Emerson  
Editing: Stella Hackell  
Cover Design: Edmonds Design

Revision History  
19 March, 2010—Revision

The information in this document is current as of the date listed in the revision history.

## END USER LICENSE AGREEMENT

**READ THIS END USER LICENSE AGREEMENT (“AGREEMENT”) BEFORE DOWNLOADING, INSTALLING, OR USING THE SOFTWARE.** BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE OR OTHERWISE EXPRESSING YOUR AGREEMENT TO THE TERMS CONTAINED HEREIN, YOU (AS CUSTOMER OR IF YOU ARE NOT THE CUSTOMER, AS A REPRESENTATIVE/AGENT AUTHORIZED TO BIND THE CUSTOMER) CONSENT TO BE BOUND BY THIS AGREEMENT. IF YOU DO NOT OR CANNOT AGREE TO THE TERMS CONTAINED HEREIN, THEN (A) DO NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND (B) YOU MAY CONTACT JUNIPER NETWORKS REGARDING LICENSE TERMS.

1. **The Parties.** The parties to this Agreement are (i) Juniper Networks, Inc. (if the Customer’s principal office is located in the Americas) or Juniper Networks (Cayman) Limited (if the Customer’s principal office is located outside the Americas) (such applicable entity being referred to herein as “Juniper”), and (ii) the person or organization that originally purchased from Juniper or an authorized Juniper reseller the applicable license(s) for use of the Software (“Customer”) (collectively, the “Parties”).

2. **The Software.** In this Agreement, “Software” means the program modules and features of the Juniper or Juniper-supplied software, for which Customer has paid the applicable license or support fees to Juniper or an authorized Juniper reseller, or which was embedded by Juniper in equipment which Customer purchased from Juniper or an authorized Juniper reseller. “Software” also includes updates, upgrades and new releases of such software. “Embedded Software” means Software which Juniper has embedded in or loaded onto the Juniper equipment and any updates, upgrades, additions or replacements which are subsequently embedded in or loaded onto the equipment.

3. **License Grant.** Subject to payment of the applicable fees and the limitations and restrictions set forth herein, Juniper grants to Customer a non-exclusive and non-transferable license, without right to sublicense, to use the Software, in executable form only, subject to the following use restrictions:

a. Customer shall use Embedded Software solely as embedded in, and for execution on, Juniper equipment originally purchased by Customer from Juniper or an authorized Juniper reseller.

b. Customer shall use the Software on a single hardware chassis having a single processing unit, or as many chassis or processing units for which Customer has paid the applicable license fees; provided, however, with respect to the Steel-Belted Radius or Odyssey Access Client software only, Customer shall use such Software on a single computer containing a single physical random access memory space and containing any number of processors. Use of the Steel-Belted Radius or IMS AAA software on multiple computers or virtual machines (e.g., Solaris zones) requires multiple licenses, regardless of whether such computers or virtualizations are physically contained on a single chassis.

c. Product purchase documents, paper or electronic user documentation, and/or the particular licenses purchased by Customer may specify limits to Customer’s use of the Software. Such limits may restrict use to a maximum number of seats, registered endpoints, concurrent users, sessions, calls, connections, subscribers, clusters, nodes, realms, devices, links, ports or transactions, or require the purchase of separate licenses to use particular features, functionalities, services, applications, operations, or capabilities, or provide throughput, performance, configuration, bandwidth, interface, processing, temporal, or geographical limits. In addition, such limits may restrict the use of the Software to managing certain kinds of networks or require the Software to be used only in conjunction with other specific Software. Customer’s use of the Software shall be subject to all such limitations and purchase of all applicable licenses.

d. For any trial copy of the Software, Customer’s right to use the Software expires 30 days after download, installation or use of the Software. Customer may operate the Software after the 30-day trial period only if Customer pays for a license to do so. Customer may not extend or create an additional trial period by re-installing the Software after the 30-day trial period.

e. The Global Enterprise Edition of the Steel-Belted Radius software may be used by Customer only to manage access to Customer’s enterprise network. Specifically, service provider customers are expressly prohibited from using the Global Enterprise Edition of the Steel-Belted Radius software to support any commercial network access services.

The foregoing license is not transferable or assignable by Customer. No license is granted herein to any user who did not originally purchase the applicable license(s) for the Software from Juniper or an authorized Juniper reseller.

4. **Use Prohibitions.** Notwithstanding the foregoing, the license provided herein does not permit the Customer to, and Customer agrees not to and shall not: (a) modify, unbundle, reverse engineer, or create derivative works based on the Software; (b) make unauthorized copies of the Software (except as necessary for backup purposes); (c) rent, sell, transfer, or grant any rights in and to any copy of the Software, in any form, to any third party; (d) remove any proprietary notices, labels, or marks on or in any copy of the Software or any product in which the Software is embedded; (e) distribute any copy of the Software to any third party, including as may be embedded in Juniper equipment sold in the secondhand market; (f) use any ‘locked’ or key-restricted feature, function, service, application, operation, or capability without first purchasing the applicable license(s) and obtaining a valid key from Juniper, even if such feature, function, service, application, operation, or capability is enabled without a key; (g) distribute any key for the Software provided by Juniper to any third party; (h) use the

Software in any manner that extends or is broader than the uses purchased by Customer from Juniper or an authorized Juniper reseller; (i) use Embedded Software on non-Juniper equipment; (j) use Embedded Software (or make it available for use) on Juniper equipment that the Customer did not originally purchase from Juniper or an authorized Juniper reseller; (k) disclose the results of testing or benchmarking of the Software to any third party without the prior written consent of Juniper; or (l) use the Software in any manner other than as expressly provided herein.

5. **Audit.** Customer shall maintain accurate records as necessary to verify compliance with this Agreement. Upon request by Juniper, Customer shall furnish such records to Juniper and certify its compliance with this Agreement.

6. **Confidentiality.** The Parties agree that aspects of the Software and associated documentation are the confidential property of Juniper. As such, Customer shall exercise all reasonable commercial efforts to maintain the Software and associated documentation in confidence, which at a minimum includes restricting access to the Software to Customer employees and contractors having a need to use the Software for Customer's internal business purposes.

7. **Ownership.** Juniper and Juniper's licensors, respectively, retain ownership of all right, title, and interest (including copyright) in and to the Software, associated documentation, and all copies of the Software. Nothing in this Agreement constitutes a transfer or conveyance of any right, title, or interest in the Software or associated documentation, or a sale of the Software, associated documentation, or copies of the Software.

8. **Warranty, Limitation of Liability, Disclaimer of Warranty.** The warranty applicable to the Software shall be as set forth in the warranty statement that accompanies the Software (the "Warranty Statement"). Nothing in this Agreement shall give rise to any obligation to support the Software. Support services may be purchased separately. Any such support shall be governed by a separate, written support services agreement. TO THE MAXIMUM EXTENT PERMITTED BY LAW, JUNIPER SHALL NOT BE LIABLE FOR ANY LOST PROFITS, LOSS OF DATA, OR COSTS OR PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, OR FOR ANY SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THIS AGREEMENT, THE SOFTWARE, OR ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. IN NO EVENT SHALL JUNIPER BE LIABLE FOR DAMAGES ARISING FROM UNAUTHORIZED OR IMPROPER USE OF ANY JUNIPER OR JUNIPER-SUPPLIED SOFTWARE. EXCEPT AS EXPRESSLY PROVIDED IN THE WARRANTY STATEMENT TO THE EXTENT PERMITTED BY LAW, JUNIPER DISCLAIMS ANY AND ALL WARRANTIES IN AND TO THE SOFTWARE (WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE), INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT DOES JUNIPER WARRANT THAT THE SOFTWARE, OR ANY EQUIPMENT OR NETWORK RUNNING THE SOFTWARE, WILL OPERATE WITHOUT ERROR OR INTERRUPTION, OR WILL BE FREE OF VULNERABILITY TO INTRUSION OR ATTACK. In no event shall Juniper's or its suppliers' or licensors' liability to Customer, whether in contract, tort (including negligence), breach of warranty, or otherwise, exceed the price paid by Customer for the Software that gave rise to the claim, or if the Software is embedded in another Juniper product, the price paid by Customer for such other product. Customer acknowledges and agrees that Juniper has set its prices and entered into this Agreement in reliance upon the disclaimers of warranty and the limitations of liability set forth herein, that the same reflect an allocation of risk between the Parties (including the risk that a contract remedy may fail of its essential purpose and cause consequential loss), and that the same form an essential basis of the bargain between the Parties.

9. **Termination.** Any breach of this Agreement or failure by Customer to pay any applicable fees due shall result in automatic termination of the license granted herein. Upon such termination, Customer shall destroy or return to Juniper all copies of the Software and related documentation in Customer's possession or control.

10. **Taxes.** All license fees payable under this agreement are exclusive of tax. Customer shall be responsible for paying Taxes arising from the purchase of the license, or importation or use of the Software. If applicable, valid exemption documentation for each taxing jurisdiction shall be provided to Juniper prior to invoicing, and Customer shall promptly notify Juniper if their exemption is revoked or modified. All payments made by Customer shall be net of any applicable withholding tax. Customer will provide reasonable assistance to Juniper in connection with such withholding taxes by promptly: providing Juniper with valid tax receipts and other required documentation showing Customer's payment of any withholding taxes; completing appropriate applications that would reduce the amount of withholding tax to be paid; and notifying and assisting Juniper in any audit or tax proceeding related to transactions hereunder. Customer shall comply with all applicable tax laws and regulations, and Customer will promptly pay or reimburse Juniper for all costs and damages related to any liability incurred by Juniper as a result of Customer's non-compliance or delay with its responsibilities herein. Customer's obligations under this Section shall survive termination or expiration of this Agreement.

11. **Export.** Customer agrees to comply with all applicable export laws and restrictions and regulations of any United States and any applicable foreign agency or authority, and not to export or re-export the Software or any direct product thereof in violation of any such restrictions, laws or regulations, or without all necessary approvals. Customer shall be liable for any such violations. The version of the Software supplied to Customer may contain encryption or other capabilities restricting Customer's ability to export the Software without an export license.

12. **Commercial Computer Software.** The Software is "commercial computer software" and is provided with restricted rights. Use, duplication, or disclosure by the United States government is subject to restrictions set forth in this Agreement and as provided in DFARS 227.7201 through 227.7202-4, FAR 12.212, FAR 27.405(b)(2), FAR 52.227-19, or FAR 52.227-14 (ALT III) as applicable.

13. **Interface Information.** To the extent required by applicable law, and at Customer's written request, Juniper shall provide Customer with the interface information needed to achieve interoperability between the Software and another independently created program, on payment of applicable fee, if any. Customer shall observe strict obligations of confidentiality with respect to such information and shall use such information in compliance with any applicable terms and conditions upon which Juniper makes such information available.

14. **Third Party Software.** Any licensor of Juniper whose software is embedded in the Software and any supplier of Juniper whose products or technology are embedded in (or services are accessed by) the Software shall be a third party beneficiary with respect to this Agreement, and such licensor or vendor shall have the right to enforce this Agreement in its own name as if it were Juniper. In addition, certain third party software may be provided with the Software and is subject to the accompanying license(s), if any, of its respective owner(s). To the extent portions of the Software are distributed under and subject to open source licenses obligating Juniper to make the source code for such portions publicly available (such as the GNU General Public License ("GPL") or the GNU Library General Public License ("LGPL")), Juniper will make such source code portions (including Juniper modifications, as appropriate) available upon request for a period of up to three years from the date of distribution. Such request can be made in writing to Juniper Networks, Inc., 1194 N. Mathilda Ave., Sunnyvale, CA 94089, ATTN: General Counsel. You may obtain a copy of the GPL at <http://www.gnu.org/licenses/gpl.html>, and a copy of the LGPL at <http://www.gnu.org/licenses/lgpl.html>.

15. **Miscellaneous.** This Agreement shall be governed by the laws of the State of California without reference to its conflicts of laws principles. The provisions of the U.N. Convention for the International Sale of Goods shall not apply to this Agreement. For any disputes arising under this Agreement, the Parties hereby consent to the personal and exclusive jurisdiction of, and venue in, the state and federal courts within Santa Clara County, California. This Agreement constitutes the entire and sole agreement between Juniper and the Customer with respect to the Software, and supersedes all prior and contemporaneous agreements relating to the Software, whether oral or written (including any inconsistent terms contained in a purchase order), except that the terms of a separate written agreement executed by an authorized Juniper representative and Customer shall govern to the extent such terms are inconsistent or conflict with terms contained herein. No modification to this Agreement nor any waiver of any rights hereunder shall be effective unless expressly assented to in writing by the party to be charged. If any portion of this Agreement is held invalid, the Parties agree that such invalidity shall not affect the validity of the remainder of this Agreement. This Agreement and associated documentation has been written in the English language, and the Parties agree that the English version will govern. (For Canada: Les parties aux présentes confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s'y rattache, soient rédigés en langue anglaise. (Translation: The parties confirm that this Agreement and all related documentation is and will be in the English language)).



# Table of Contents

	<b>About This Guide</b> .....	<b>xv</b>
	Objectives .....	xv
	Audience .....	xv
	Conventions .....	xv
	Documentation .....	xvii
	Requesting Technical Support .....	xvii
	Self-Help Online Tools and Resources .....	xviii
	Opening a Case with JTAC .....	xviii
<b>Part 1</b>	<b>NSM API</b>	
<b>Chapter 1</b>	<b>Overview</b> .....	<b>3</b>
	NSM API Features .....	3
	NSM API Authentication and Authorization .....	4
	NSM API Error Handling .....	4
<b>Chapter 2</b>	<b>NSM API Operations</b> .....	<b>7</b>
	System Service API .....	7
	Data Centric Service API .....	8
	Data Centric Service XML Subtree Filter .....	8
	Data Centric Service Operations .....	9
	Job Service API .....	11
	Log Service API .....	13
<b>Part 2</b>	<b>API Data Types</b>	
<b>Chapter 3</b>	<b>Data Objects</b> .....	<b>17</b>
	API Data Objects .....	17
<b>Chapter 4</b>	<b>Common Message Data Types</b> .....	<b>21</b>
	SimpleRequestType and SimpleResponseType Data Types .....	21
<b>Chapter 5</b>	<b>Security Data Model</b> .....	<b>23</b>
	NSM Policy .....	23
	Security Rulebases .....	25
	Backdoor (rb_backdoor_collection) .....	25
	Exempt (rb_exempt_collection) .....	30
	Firewall (rb_firewall_collection) .....	33
	IDP (rb_idp_collection) .....	39
	Multicast (rb_multicast_collection) .....	43
	SYN Protector (rb_syndef_collection) .....	45
	Traffic Anomalies (rb_tsig_collection) .....	48

	Network Honeypot (rb_portfaker_collection) . . . . .	51
	Service (service_collection) . . . . .	54
	Address (address_collection_type) . . . . .	56
	Schedule Object (scheduleobj_collection_type) . . . . .	57
	Attack (attack_collection) . . . . .	58
	Antivirus (avobj_collection) . . . . .	62
	GTP (gtpobj_collection_type) . . . . .	64
	DI Profile (DIProfile_collection_type) . . . . .	67
	Global DIP (globaldip_collection) . . . . .	67
	Global MIP (globalmpi_collection) . . . . .	68
	Global VIP (globalvip_collection) . . . . .	69
	URL Filter Object (urlfilter_collection) . . . . .	70
<b>Part 3</b>	<b>Using the NSM API from a Perl Client</b>	
<b>Chapter 6</b>	<b>Installing the Perl Client Environment . . . . .</b>	<b>75</b>
	Installing the Perl Client Environment on Linux-Unix Machines . . . . .	75
	Installing the Perl Client Environment on Windows Machines . . . . .	76
	Using a Perl Script to Access the NSM API . . . . .	77
<b>Chapter 7</b>	<b>Using the Perl Client to Access the NSM API . . . . .</b>	<b>81</b>
	Login and Logout . . . . .	81
<b>Chapter 8</b>	<b>Using the API to Manage Shared Objects . . . . .</b>	<b>83</b>
	Using the Perl Client Library with Address Objects . . . . .	83
	Add Address Objects . . . . .	83
	Replace an Address Object . . . . .	85
	Rename Address Objects . . . . .	85
	Read Address Objects . . . . .	86
	Delete Address Objects . . . . .	87
	Delete All Address Objects . . . . .	87
	Using the Perl Client Library with Service Objects . . . . .	87
	Add Service Objects . . . . .	88
	Add Group-Global Service Objects . . . . .	89
	Read Group-Global Service Objects . . . . .	90
	Replace Group-Global Service Objects . . . . .	91
	Delete All Group-Global Service Objects . . . . .	91
	Using the Perl Client Library with Device Objects . . . . .	91
	Read Device Objects . . . . .	91
<b>Part 4</b>	<b>Using the NSM API from a Java Client</b>	
<b>Chapter 9</b>	<b>Using APIs for Authentication . . . . .</b>	<b>95</b>
	Login . . . . .	95
	Logout . . . . .	96
<b>Chapter 10</b>	<b>Using APIs for Policy Management . . . . .</b>	<b>97</b>
	Create a New Policy . . . . .	97
	Update an Existing Policy . . . . .	99
	Delete a Policy . . . . .	100
	Get a List of Policies . . . . .	101

	Get a Policy . . . . .	101
	Assign a Policy to a Device . . . . .	102
	Remove a Policy Assignment . . . . .	103
<b>Chapter 11</b>	<b>Using APIs for Shared Object Management . . . . .</b>	<b>105</b>
	Insert a Shared Object . . . . .	105
	Replace a Shared Object . . . . .	106
	Delete a Shared Object . . . . .	108
	Get a List of Shared Objects . . . . .	108
	Get a Shared Object . . . . .	109
<b>Chapter 12</b>	<b>Using APIs for Job Management . . . . .</b>	<b>111</b>
	Get a Job Result . . . . .	111
	Import a List of Devices . . . . .	112
	Update a List of Devices . . . . .	113
	Get a Configuration Summary . . . . .	113
	Get a Running Configuration . . . . .	114
	Get the Delta Configuration . . . . .	115
	Cancel a Job Request . . . . .	115
<b>Chapter 13</b>	<b>Using APIs for Device Management . . . . .</b>	<b>117</b>
	Retrieve Domains . . . . .	117
	Retrieve the Device List in One Domain . . . . .	117
<b>Part 5</b>	<b>NSM API WSDLs</b>	
<b>Chapter 14</b>	<b>Job Service API WSDL . . . . .</b>	<b>121</b>
	WSDL File . . . . .	121
<b>Chapter 15</b>	<b>System Service API WSDL . . . . .</b>	<b>129</b>
	WSDL File . . . . .	129
<b>Chapter 16</b>	<b>Data Centric API WSDL . . . . .</b>	<b>135</b>
	WSDL File . . . . .	135
<b>Chapter 17</b>	<b>Log Service API WSDL . . . . .</b>	<b>145</b>
	WSDL File . . . . .	145
<b>Part 6</b>	<b>Index</b>	
	Index . . . . .	149



# List of Figures

<b>Part 1</b>	<b>NSM API</b>	
<b>Chapter 1</b>	<b>Overview</b> .....	<b>3</b>
	Figure 1: ErrorType Data Type .....	5
<b>Part 2</b>	<b>API Data Types</b>	
<b>Chapter 3</b>	<b>Data Objects</b> .....	<b>17</b>
	Figure 2: NSM API Data Objects .....	18
<b>Chapter 4</b>	<b>Common Message Data Types</b> .....	<b>21</b>
	Figure 3: SimpleRequestType Data Type .....	21
	Figure 4: SimpleResponseType Data Type .....	21
<b>Chapter 5</b>	<b>Security Data Model</b> .....	<b>23</b>
	Figure 5: NSM Policy .....	24
	Figure 6: Backdoor Rulebase .....	27
	Figure 7: Exempt Rulebase .....	31
	Figure 8: Firewall Rulebase .....	33
	Figure 9: Firewall policy_type .....	34
	Figure 10: IDP Rulebase .....	40
	Figure 11: Multicast Rulebase .....	44
	Figure 12: SYN Protector Rulebase .....	46
	Figure 13: Traffic Anomalies Rulebase .....	49
	Figure 14: Network Honeypot Rulebase .....	52
	Figure 15: Service Collection .....	55
	Figure 16: Address Collection .....	57
	Figure 17: Schedule Object .....	58
	Figure 18: Antivirus Collection .....	63
	Figure 19: GTP Collection .....	65
	Figure 20: DI Profile .....	67
	Figure 21: Global DIP Collection .....	68
	Figure 22: Global MIP Collection .....	69
	Figure 23: Global VIP Collection .....	70
	Figure 24: URL Filter Object Collection .....	71



# List of Tables

	<b>About This Guide</b> . . . . .	<b>xv</b>
	Table 1: Text Conventions . . . . .	xvi
	Table 2: Syntax Conventions . . . . .	xvi
	Table 3: Network and Security Manager Publications . . . . .	xvii
<b>Part 1</b>	<b>NSM API</b>	
<b>Chapter 1</b>	<b>Overview</b> . . . . .	<b>3</b>
	Table 4: ErrorType Data Types . . . . .	5
<b>Chapter 2</b>	<b>NSM API Operations</b> . . . . .	<b>7</b>
	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
<b>Part 2</b>	<b>API Data Types</b>	
<b>Chapter 3</b>	<b>Data Objects</b> . . . . .	<b>17</b>
	Table 9: API Data Objects . . . . .	18
<b>Chapter 4</b>	<b>Common Message Data Types</b> . . . . .	<b>21</b>
	Table 10: SimpleRequestType and SimpleResponseType Definitions . . . . .	22
<b>Chapter 5</b>	<b>Security Data Model</b> . . . . .	<b>23</b>
	Table 11: NSM Policy Data Elements . . . . .	24
	Table 12: Backdoor Rulebase Data Elements . . . . .	27
	Table 13: Exempt Rulebase Data Elements . . . . .	31
	Table 14: Firewall Data Elements . . . . .	35
	Table 15: IDP Rulebase Data Elements . . . . .	41
	Table 16: Multicast Rulebase Data Elements . . . . .	44
	Table 17: SYN Protector Rulebase Data Elements . . . . .	46
	Table 18: Traffic Anamolies Rulebase Date Elements . . . . .	49
	Table 19: Network Honeypot Rulebase Data Elements . . . . .	52
	Table 20: Service Collection Data Elements . . . . .	55
	Table 21: Address Collection Data Elements . . . . .	57
	Table 22: Schedule Object Data Elements . . . . .	58
	Table 23: Attack Collection Data Elements . . . . .	61
	Table 24: Antivirus Collection Data Elements . . . . .	63
	Table 25: GTP Collection Data Elements . . . . .	65
	Table 26: DIP Data Elements . . . . .	67
	Table 27: Global DIP Data Elements . . . . .	68

Table 28: Global MIP Data Elements . . . . . 69  
Table 29: Global VIP Data Elements . . . . . 70  
Table 30: URL Filter Data Collection . . . . . 71

# 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 xv
- Audience on page xv
- Conventions on page xv
- Documentation on page xvii
- Requesting Technical Support on page xvii

## 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 xvi defines text conventions used in this guide.

Table 1: Text Conventions

Convention	Description	Examples
<b>Bold typeface like this</b>	<ul style="list-style-type: none"> <li>Represents commands and keywords in text.</li> <li>Represents keywords</li> <li>Represents UI elements</li> </ul>	<ul style="list-style-type: none"> <li>Issue the <b>clock source</b> command.</li> <li>Specify the keyword <b>exp-msg</b>.</li> <li>Click <b>User Objects</b></li> </ul>
<b>Bold typeface like this</b>	Represents text that the user must type.	<b>user input</b>
<code>fixed-width font</code>	Represents information as displayed on the terminal screen.	<pre>host1# show ip ospf Routing Process OSPF 2 with Router ID 5.5.0.250 Router is an area Border Router (ABR)</pre>
Key names linked with a plus (+) sign	Indicates that you must press two or more keys simultaneously.	Ctrl + d
<i>Italics</i>	<ul style="list-style-type: none"> <li>Emphasizes words</li> <li>Identifies variables</li> </ul>	<ul style="list-style-type: none"> <li>The product supports two levels of access, <i>user</i> and <i>privileged</i>.</li> <li><i>clusterID</i>, <i>ipAddress</i>.</li> </ul>
The angle bracket (>)	Indicates navigation paths through the UI by clicking menu options and links.	<b>Object Manager &gt; User Objects &gt; Local Objects</b>

Table 2 on page xvi 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	<i>mask</i> , <i>accessListName</i>
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 xvii describes documentation for the NSM.

**Table 3: Network and Security Manager Publications**

Book	Description
<i>Network and Security Manager Installation Guide</i>	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.
<i>Network and Security Manager Administration Guide</i>	<p>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.</p> <p>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.</p>
<i>Network and Security Manager Configuring Screen OS and IDP Devices Guide</i>	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.
<i>Network and Security Manager Online Help</i>	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.
<i>Network and Security Manager API Guide</i>	Provides complete syntax and description of the SOAP messaging interface to NSM.
<i>Network and Security Manager Release Notes</i>	<p>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.</p> <p>Release notes are included on the corresponding software CD and are available on the Juniper Networks Website.</p>

## 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: <https://www.juniper.net/alerts/>
- 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

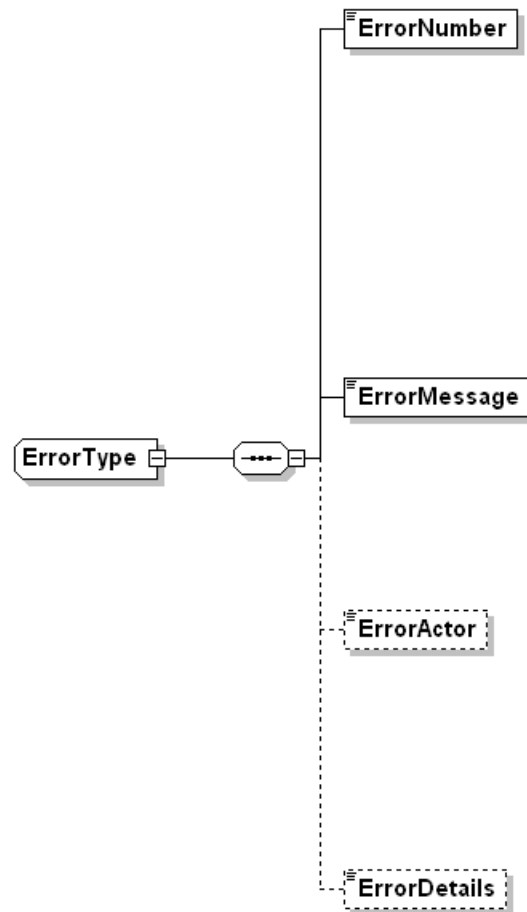


Table 4: ErrorType Data Types

Data Type	Description
ErrorType	<p>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:</p> <ul style="list-style-type: none"> <li>• ErrorNumber = Unique number that identifies the particular error condition (type = unsignedInt). This data element is only used by the server.</li> <li>• ErrorMessage = Brief description of the condition that raised the error (type = string).</li> <li>• ErrorActor = The source (location) of the error (type = string).</li> <li>• ErrorDetails = Detailed error message (type = string).</li> </ul>



## 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 129 .

**Table 5: System Service API Operations**

Operation	Description
LoginRequest	<p>Log into NSM server.</p> <p>Request:</p> <ul style="list-style-type: none"><li>• domainName = Domain supplied during login. The user logs in to this domain.</li></ul> <p><b>NOTE:</b> Use <b>global.&lt;subdomain name&gt;</b> to log in to a subdomain and <b>global</b> to log in to a global domain.</p> <ul style="list-style-type: none"><li>• userName = User name supplied during login.</li><li>• password = Password supplied during login.</li></ul> <p>Response:</p> <ul style="list-style-type: none"><li>• loginStatus = Uses LoginStatusCodeType to return:<ul style="list-style-type: none"><li>“Success” if the login is successful.</li><li>“Failure” for login rejection.</li><li>“Challenge” if the login request is being challenged but not yet denied.</li></ul></li><li>• authToken = Token returned for login request success. This token is reused for other requests during the current session.</li></ul>

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.



**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 135 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	<p>Gets objects that refer to the object specified in the request.</p> <p>Request:</p> <ul style="list-style-type: none"> <li>• objectIdentifier= Identifies the object to be retrieved (type = objectIdentifierType)</li> <li>• dbVersionId= Version of the database (type = unsignedInt)</li> <li>• objectFilter = Filter to be applied to the result (type = ObjectFilterType)</li> <li>• metadataOnly = If true, only the metadata is returned. Otherwise, the entire object is returned.</li> </ul> <p>Response: object</p>
GetObjectViewByCategoryRequest	<p>Gets objects in one category.</p> <p>Request:</p> <ul style="list-style-type: none"> <li>• category = Schema name of the category (type = string)</li> <li>• domainId = Domain of the schema (type = unsignedShort)</li> <li>• dbVersionId= Version of the database (type = unsignedInt)</li> <li>• objectFilter = Filter to be applied to the result (type = ObjectFilterType)</li> <li>• view = Transformation of the object. For the default view, the returned object follows the schema with no transformation (type = ViewType).</li> <li>• property = Transformation parameters (type = NameValueType).</li> </ul> <p>Response: object</p>

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

GetObjectViewByIdRequest	<p>Gets objects by ID.</p> <p>Request:</p> <ul style="list-style-type: none"> <li>• objectIdentifier= Identifies the object to be retrieved (type = objectIdentifierType)</li> <li>• dbVersionId= Version of the database (type = unsignedInt)</li> <li>• objectFilter = Filter to be applied to the result (type = ObjectFilterType)</li> <li>• view = Transformation of the object. For the default view, the returned object follows the schema with no transformation (type = ViewType).</li> <li>• property = Transformation parameters (type = NameValueType).</li> </ul> <p>Response: object</p>
LockObjectRequest	<p>Locks the specified object.</p> <p>Request: objectIdentifier</p> <p>Response: objectLockStatus</p>
UnlockObjectRequest	<p>Unlocks the specified object.</p> <p>Request: object</p> <p>Response: objectLockStatus</p>
ModifyObjectViewRequest	<p>Modifies the object. All commands in the request are executed in one transaction. ModifyObjectViewRequest supports the following operations:</p> <ul style="list-style-type: none"> <li>• Update Node</li> <li>• Insert node before / after</li> <li>• Append node</li> <li>• Insert Object</li> <li>• Replace Object</li> <li>• Delete Object</li> </ul> <p><b>NOTE:</b> 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.</p> <p>Request:</p> <ul style="list-style-type: none"> <li>• command = Command that modifies the object (type = ModifyCommand).</li> </ul> <p>Response:</p> <ul style="list-style-type: none"> <li>• metadata</li> <li>• objectModification</li> <li>• subObjectModification</li> </ul>

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

QueryObjectViewRequest	<p>Queries the object.</p> <p>Request:</p> <ul style="list-style-type: none"> <li>category = Schema name of the category (type = string).</li> <li>simpleQuery = Query expression (type = SimpleQueryType).</li> <li>dbVersionId= Version of the database (type = unsignedInt)</li> <li>objectFilter = Filter to be applied to the result (type = ObjectFilterType)</li> <li>view = Transformation of the object. For the default view, the returned object follows the schema with no transformation (type = ViewType).</li> <li>property = Transformation parameters (type = NameValueType).</li> </ul> <p>Response:</p> <ul style="list-style-type: none"> <li>queryId</li> <li>object</li> </ul>
ResolveObjectReferenceRequest	<p>Resolves the object reference.</p> <p>Request:</p> <ul style="list-style-type: none"> <li>objectReference = Object reference to be resolved (type = string)</li> <li>dbVersionId= Version of the database (type = unsignedInt)</li> <li>objectFilter = Filter to be applied to the result (type = ObjectFilterType)</li> </ul> <p>Response: object</p>

## 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 121 for the Web Service Description Language (WSDL) definition of the API.

Table 7: Job Service API Operations

Operation	Description
UpdateDeviceRequest	<p>Request to update the device configuration.</p> <p>Request: JobRequestType</p> <ul style="list-style-type: none"> <li>jobName= Name of the job.</li> <li>scheduleTime= Time when the job will run. If not specified, the job will run immediately.</li> <li>jobArgs= List of the devices to which the job applies (type = JobArgsType).</li> </ul> <p>Response: JobResponseType</p> <ul style="list-style-type: none"> <li>status = Job status.</li> <li>jobName = Name of the job.</li> <li>response = Response to the job.</li> </ul>

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

Operation	Description
ImportDeviceRequest	<p>Request to import a device configuration from physical devices.</p> <p>Request: JobRequestType</p> <ul style="list-style-type: none"> <li>• jobName= Name of the job.</li> <li>• scheduleTime= Time when the job will run. If not specified, the job will run immediately.</li> <li>• jobArgs= List of the devices to which the job applies.</li> </ul> <p>Response: JobResponseType</p> <ul style="list-style-type: none"> <li>• status = Job status.</li> <li>• jobName = Name of the job.</li> <li>• response = Response to the job.</li> </ul>
GetConfigSummaryRequest	<p>Request for a summary of the configuration currently running on a physical device.</p> <p>Request: JobRequestType</p> <ul style="list-style-type: none"> <li>• jobName= Name of the job.</li> <li>• scheduleTime= Time when the job will run. If not specified, the job will run immediately.</li> <li>• jobArgs= List of the devices to which the job applies.</li> </ul> <p>Response: JobResponseType</p> <ul style="list-style-type: none"> <li>• status = Job status.</li> <li>• jobName = Name of the job.</li> <li>• response = Response to the job.</li> </ul>
GetRunningConfigRequest	<p>Request for the configuration currently running on a physical device.</p> <p>Request: JobRequestType</p> <ul style="list-style-type: none"> <li>• jobName= Name of the job.</li> <li>• scheduleTime= Time when the job will run. If not specified, the job will run immediately.</li> <li>• jobArgs= List of the devices to which the job applies.</li> </ul> <p>Response: JobResponseType</p> <ul style="list-style-type: none"> <li>• status = Job status.</li> <li>• jobName = Name of the job.</li> <li>• response = Response to the job.</li> </ul>

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

Operation	Description
GetDeltaConfigRequest	<p>Request for the differences between the modeled device configuration and the configuration currently running on a physical device.</p> <p>Request: JobRequestType</p> <ul style="list-style-type: none"> <li>• jobName= Name of the job.</li> <li>• scheduleTime= Time when the job will run. If not specified, the job will run immediately.</li> <li>• jobArgs= List of the devices to which the job applies.</li> </ul> <p>Response: JobResponseType</p> <ul style="list-style-type: none"> <li>• status = Job status.</li> <li>• jobName = Name of the job.</li> <li>• response = Response to the job.</li> </ul>
GetJobStatusRequest	<p>Request for the status of a job.</p> <p>Request:</p> <ul style="list-style-type: none"> <li>• domainName = Name of the domain associated with the job.</li> <li>• domainId= ID of the domain.</li> <li>• jobName= Name of the job.</li> </ul> <p>Response: JobResponseType</p> <ul style="list-style-type: none"> <li>• status = Job status.</li> <li>• jobName = Name of the job.</li> <li>• response = Response to the job.</li> </ul>
GetJobResultRequest	<p>Request for status of a completed job.</p> <p>Request:</p> <ul style="list-style-type: none"> <li>• domainName = Name of the domain associated with the job.</li> <li>• domainId= ID of the domain.</li> <li>• jobName= Name of the job.</li> </ul> <p>Response: JobResponseType</p> <ul style="list-style-type: none"> <li>• status = Job status.</li> <li>• jobName = Name of the job.</li> <li>• response = Response to the job.</li> </ul>

## 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 135 for the WSDL file defining the API.

**Table 8: Log Service API Operations**

Operation	Description
GetPacketDataRequest	<p>Gets both the log data and the packet data that triggers the log.</p> <p>Request:</p> <ul style="list-style-type: none"><li>• dayId= Identifier for the day.</li><li>• recordNum= Record number.</li></ul> <p>Response:</p> <ul style="list-style-type: none"><li>• numPackets= Number of packets returned.</li><li>• triggerPacket = Packet triggering the log event.</li><li>• data = Log data.</li></ul>

## 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

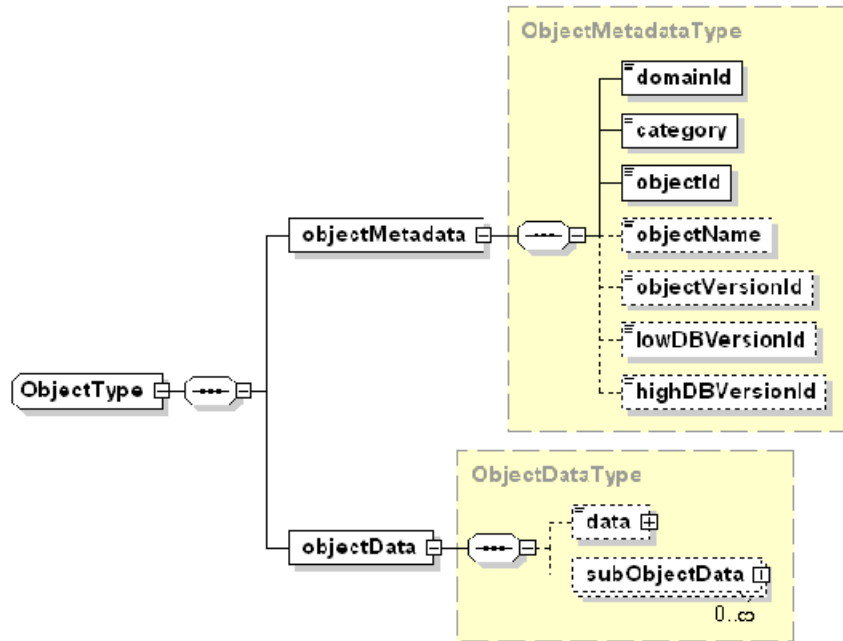


Table 9: API Data Objects

Data Type	Description
DataFormatType	<p>This simpleType data code has the following possible enumeration values:</p> <ul style="list-style-type: none"> <li>• "XML"</li> <li>• "NML"</li> <li>• "XML_FROM_NML"</li> <li>• "NML_AND_XML"</li> <li>• "JAVA_OBJECT"</li> <li>• "FILE"</li> </ul>
OpaqueDataType	<p>This complexType data code (base64Binary element) has one value:</p> <ul style="list-style-type: none"> <li>• attribute name = "dataFormat" (type = DataFormatType)</li> </ul>
ObjectIdentifierType	<p>This complexType data code has the following sequence:</p> <ul style="list-style-type: none"> <li>• domainId = ID of the domain (type = unsignedShort)</li> <li>• category = Schema name of the category (type = string)</li> <li>• objectIdOrName = Object ID or name (type = ObjectIdOrNameType).</li> </ul>

Table 9: API Data Objects (*continued*)

Data Type	Description
ObjectMetadataType	This complexType data code has the following sequence: <ul style="list-style-type: none"> <li>domainId = ID of the domain (type = unsignedShort)</li> <li>category = Schema name of the category (type = string)</li> <li>objectId = Object identifier (type = unsignedInt).</li> <li>objectName = Name of the object (type = string).</li> <li>objectVersionId = Identifier of the object version (type = unsignedInt).</li> <li>lowDBVersionId= Lowest database version identifier (type = unsignedInt).</li> <li>highDBVersionId= Highest database version identifier (type = unsignedInt).</li> </ul>
ObjectIdOrNameType	This complexType data code takes only one of the following inputs: <ul style="list-style-type: none"> <li>objectId = Object identifier (type = unsignedInt).</li> <li>or</li> <li>objectName = Name of the object (type = string).</li> </ul>
DomainIdOrNameType	This complexType data code takes only one of the following inputs: <ul style="list-style-type: none"> <li>objectId = Object identifier (type = unsignedInt).</li> <li>or</li> <li>objectName = Name of the object (type = string).</li> </ul>
SubObjectType	This complexType data code has the following sequence: <ul style="list-style-type: none"> <li>subCategory = Subcategory under "category" (type = string).</li> <li>data = Subobject data (type = OpaqueDataType).</li> </ul>
ObjectType	This complexType data code has the following sequence: <ul style="list-style-type: none"> <li>objectName = Name of the object (type = string).</li> <li>data = Object data (type = OpaqueDataType).</li> <li>subObjectData = Data of the subobject (type = SubObjectType).</li> </ul>
ObjectDataType	This complexType data code has the following sequence: <ul style="list-style-type: none"> <li>objectName = Name of the object (type = string).</li> <li>data = Object data (type = OpaqueDataType).</li> <li>subObjectData = Data of the subobject (type = SubObjectType).</li> </ul>
ObjectType	This complexType data code has the following sequence: <ul style="list-style-type: none"> <li>objectMetadata = Metadata object (type = ObjectMetadataType).</li> <li>objectData = Object data (type = ObjectDataType).</li> </ul>
StatusCodeType	This simpleType data code has the following possible enumeration values: <ul style="list-style-type: none"> <li>Success = Request is successful.</li> <li>Failure = Request has failed.</li> </ul>
AuthTokenType	This complexType data code is the security header for SOAP API calls. It has one value: <ul style="list-style-type: none"> <li>Token = String identifying the user (type = string).</li> </ul>
SequenceType	This complexType data code is a partial response type. It has the following sequence: <ul style="list-style-type: none"> <li>SequenceNum= Sequence number of the current response (type = int).</li> <li>IsDone = Total number of response messages (type = Boolean).</li> </ul>

**Table 9: API Data Objects (continued)**

Data Type	Description
ProgressType	This complexType data code takes one input: <ul style="list-style-type: none"><li>CompletionPercent = Percent completed of the response (type = unsignedInt).</li></ul>
ConversationContextType	This complexType data code has the following sequence: <ul style="list-style-type: none"><li>ConversationId = Identifier for the message conversation (type = string).</li><li>UserSessionContext = Describes the context of the user session (type = anyType).</li><li>AuditLogContext = Context for the audit log (type = anyType).</li><li>ACFilter = Filter (type = anyType).</li></ul>

# 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

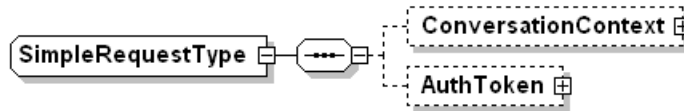


Figure 4: SimpleResponseType Data Type

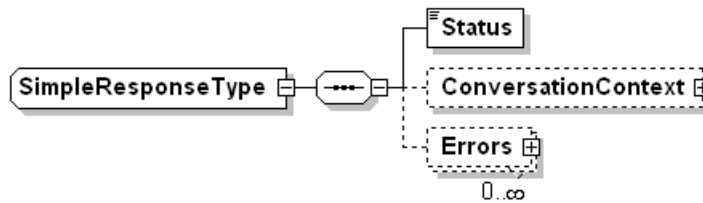


Table 10: SimpleRequestType and SimpleResponseType Definitions

Data Type	Description
SimpleRequestType	<p>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.</p> <p>This complexType data has the following sequence:</p> <ul style="list-style-type: none"><li>• ConversationContext = Context of the message conversation (type = ConversationContextType).</li><li>• AuthToken = Token returned for the simple request (type = AuthTokenType).</li></ul>
SimpleResponseType	<p>Base type definition of the SOAP body of a response. This complexType data has the following sequence:</p> <ul style="list-style-type: none"><li>• Status = Status of the response (type = StatusCodeType).</li><li>• ConversationContext = Context of the message conversation (type = ConversationContextType).</li><li>• Errors = Errors returned (type = ErrorType).</li></ul>

## 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 23
- Security Rulebases on page 25
- Service (`service_collection`) on page 54
- Address (`address_collection_type`) on page 56
- Schedule Object (`scheduleobj_collection_type`) on page 57
- Attack (`attack_collection`) on page 58
- Antivirus (`avobj_collection`) on page 62
- GTP (`gtpobj_collection_type`) on page 64
- DI Profile (`DIPprofile_collection_type`) on page 67
- Global DIP (`globaldip_collection`) on page 67
- Global MIP (`globalmpi_collection`) on page 68
- Global VIP (`globalvip_collection`) on page 69
- URL Filter Object (`urlfilter_collection`) on page 70

## 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

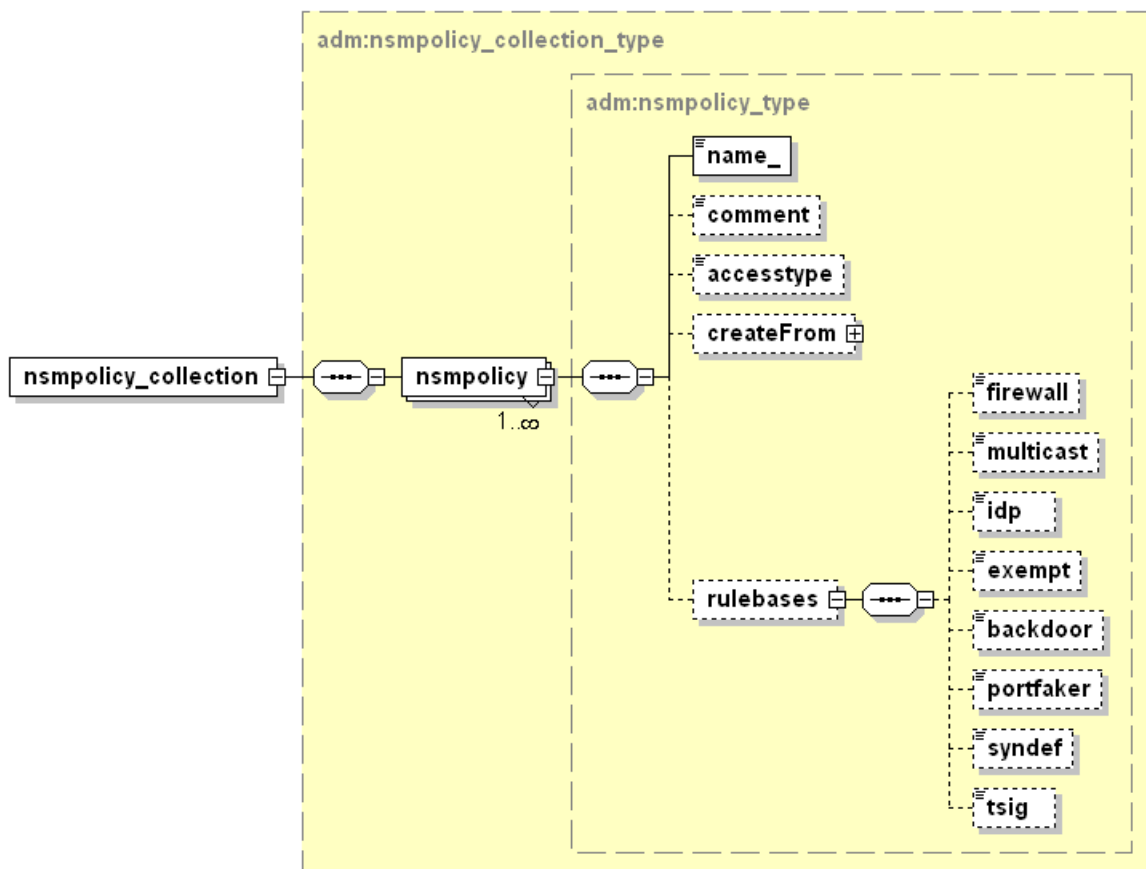


Table 11: NSM Policy Data Elements

Data Element	Description
name_	Name of the security policy (string).
comment	Comments about the security policy.
accesstype	Type of access. (enum) Possible values are: <ul style="list-style-type: none"> <li>regular = regular policy</li> <li>pre = domain pre policy</li> <li>post = domain post policy</li> <li>mompre = central manager pre policy</li> <li>mompost = central manager post policy</li> </ul>
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.
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 33.

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

Data Element	Description
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 43.
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 39.
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 30.
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 48.
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 45.
tsig	Traffic Anomalies rulebase. These data elements are included in a security policy. For more information, see “Traffic Anomalies (rb_tsig_collection)” on page 48.

## 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 27 and Table 12 on page 27.

Figure 6: Backdoor Rulebase

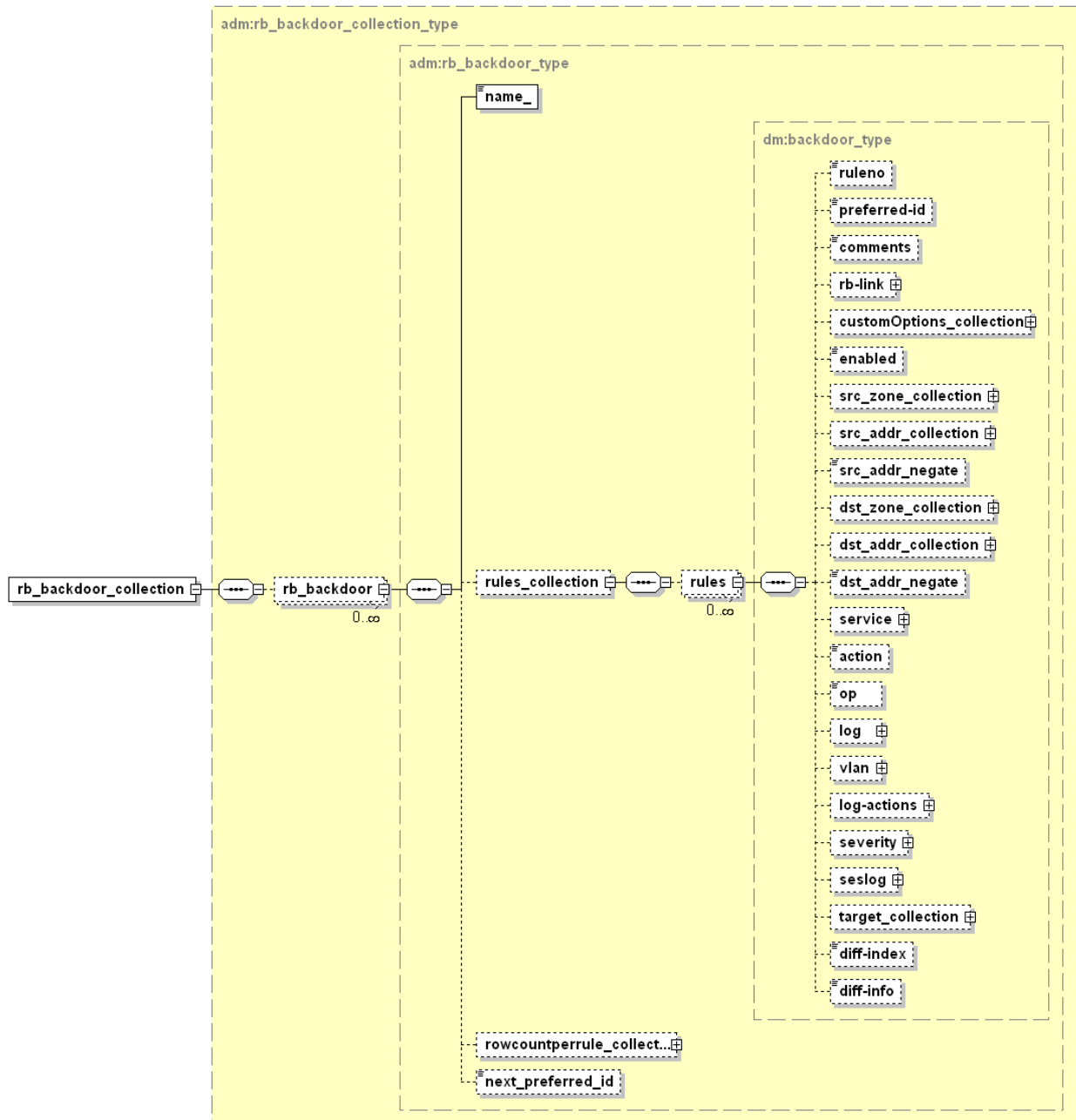


Table 12: Backdoor Rulebase Data Elements

Data Element	Description
name_	Name of the backdoor rule type. (string).
rules_collection	Collection of all sets of rules.
rules	Collection of all rules.

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

Data Element	Description
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	<p>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:</p> <ul style="list-style-type: none"> <li>• Accept = IDP accepts the interactive traffic</li> <li>• 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.</li> <li>• Close Client = IDP closes the interactive connection to the client but not to the server.</li> <li>• Close Server = IDP closes the interactive connection to the server but not to the client.</li> <li>• 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.</li> </ul>
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	<p>This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> </ul>
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: <ul style="list-style-type: none"> <li>• Default</li> <li>• Info</li> <li>• Warning</li> <li>• Minor</li> <li>• Major</li> <li>• Critical</li> </ul>
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 31 and Table 13 on page 31.

Figure 7: Exempt Rulebase

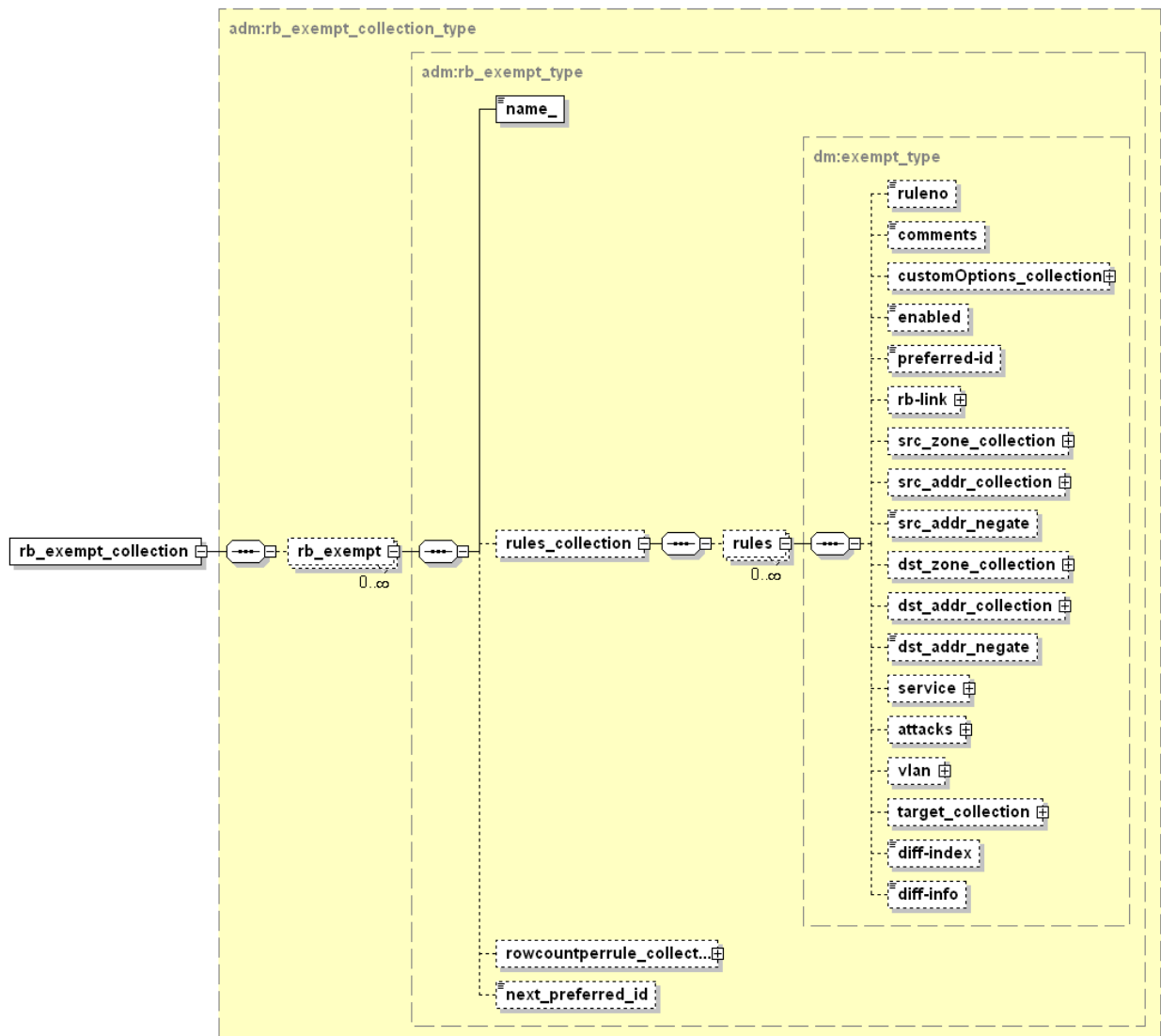


Table 13: Exempt Rulebase Data Elements

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.

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

Data Element	Description
rueno	Rule number.
comments	Comments about the exempt 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	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: <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> <li>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.</li> </ul>
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 33, Figure 9 on page 34, and Table 14 on page 35.

Figure 8: Firewall Rulebase

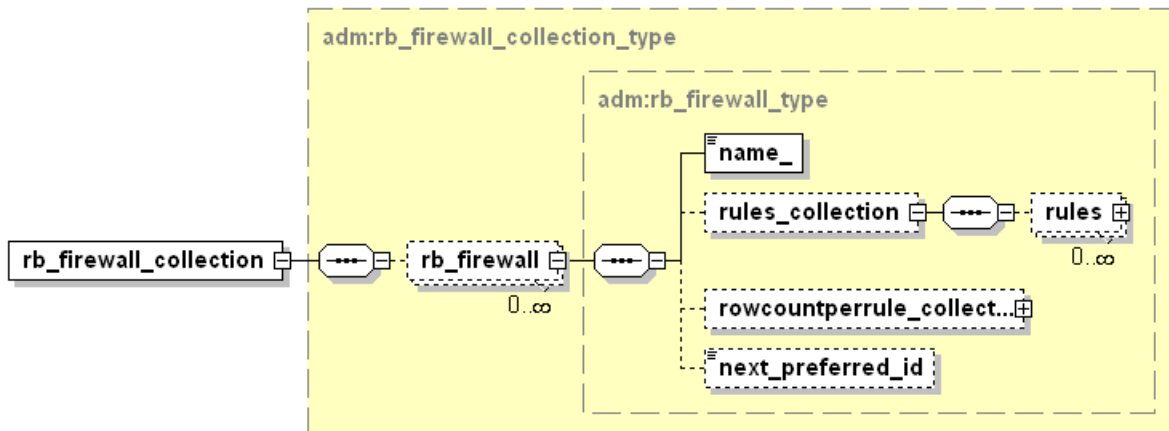


Figure 9: Firewall policy\_type

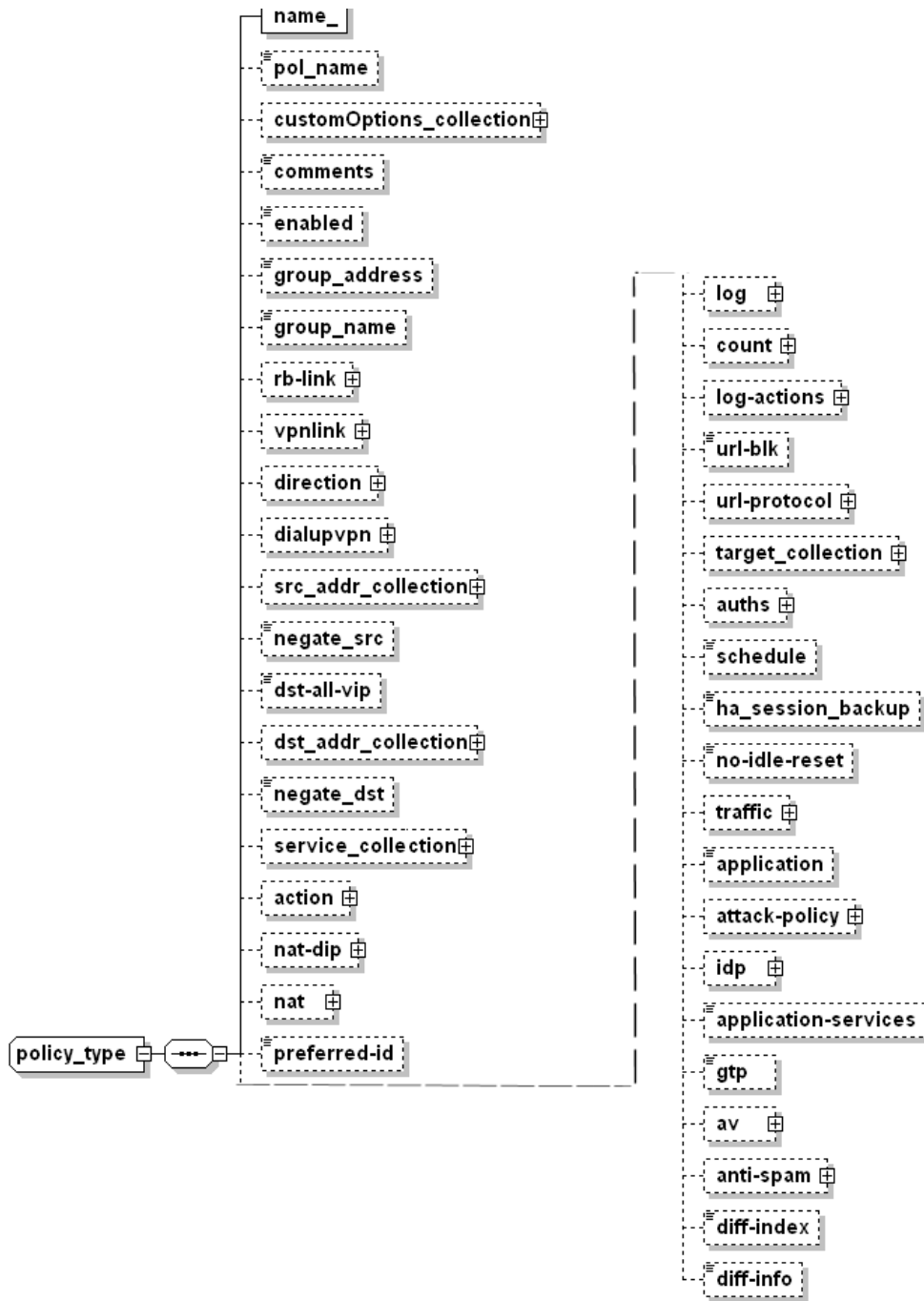


Table 14: Firewall Data Elements

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 54.
action	Determines the action to be performed by the security device when it detects traffic that matches the rule. The possible values are: <ul style="list-style-type: none"> <li>• deny</li> <li>• permit</li> <li>• reject</li> <li>• tunnel</li> </ul>

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: <ul style="list-style-type: none"> <li>disabled</li> <li>enabled</li> </ul>
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	<p>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.</p> <p>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.</p> <p>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.</p> <p>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:</p> <ul style="list-style-type: none"> <li>• no-auth</li> <li>• infranet-auth</li> <li>• auth</li> <li>• webauth</li> </ul>
schedule	<p>You can determine when a security device applies a rule to network traffic by defining a schedule for the rule.</p>
ha_session_backup	<p>If you select <b>HA Session Backup</b>, 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.</p>
no-idle-reset	<p>Disable modem idle timer reset. (default = false)</p>
traffic	<p>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:</p> <ul style="list-style-type: none"> <li>• gbw</li> <li>• priority</li> <li>• mbw</li> </ul>

Table 14: Firewall Data Elements (*continued*)

Data Element	Description
application	<p>Application.</p> <p><b>NOTE:</b> 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.</p> <p>Possible enumeration values:</p> <ul style="list-style-type: none"> <li>• DNS</li> <li>• FTP</li> <li>• HTTP</li> <li>• IMAP</li> <li>• SMTP</li> <li>• POP3</li> <li>• H245</li> <li>• Q931</li> <li>• RAS</li> <li>• PORTMAPPER</li> <li>• SIP</li> <li>• SQLNETV2</li> <li>• TALK</li> <li>• TFTP</li> <li>• REAL</li> <li>• RTSP</li> <li>• VDO</li> <li>• XING</li> <li>• IGNORE</li> <li>• MGCP_CA</li> <li>• MGCP_UA</li> <li>• PPTP</li> <li>• RSH</li> <li>• SCCP</li> <li>• AIM</li> <li>• YMSG</li> <li>• SMB</li> <li>• MSN</li> <li>• NBNAME</li> <li>• NBDS</li> <li>• NAS</li> <li>• NONE = No application specified. (default)</li> </ul>
attack-policy	<p>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.</p>
idp	<p>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.</p>

Table 14: Firewall Data Elements (*continued*)

Data Element	Description
application-services	Application services. Possible values: <ul style="list-style-type: none"> <li>• None (default)</li> <li>• RWX</li> <li>• RRWX</li> </ul>
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 40 and Table 15 on page 41.

Figure 10: IDP Rulebase

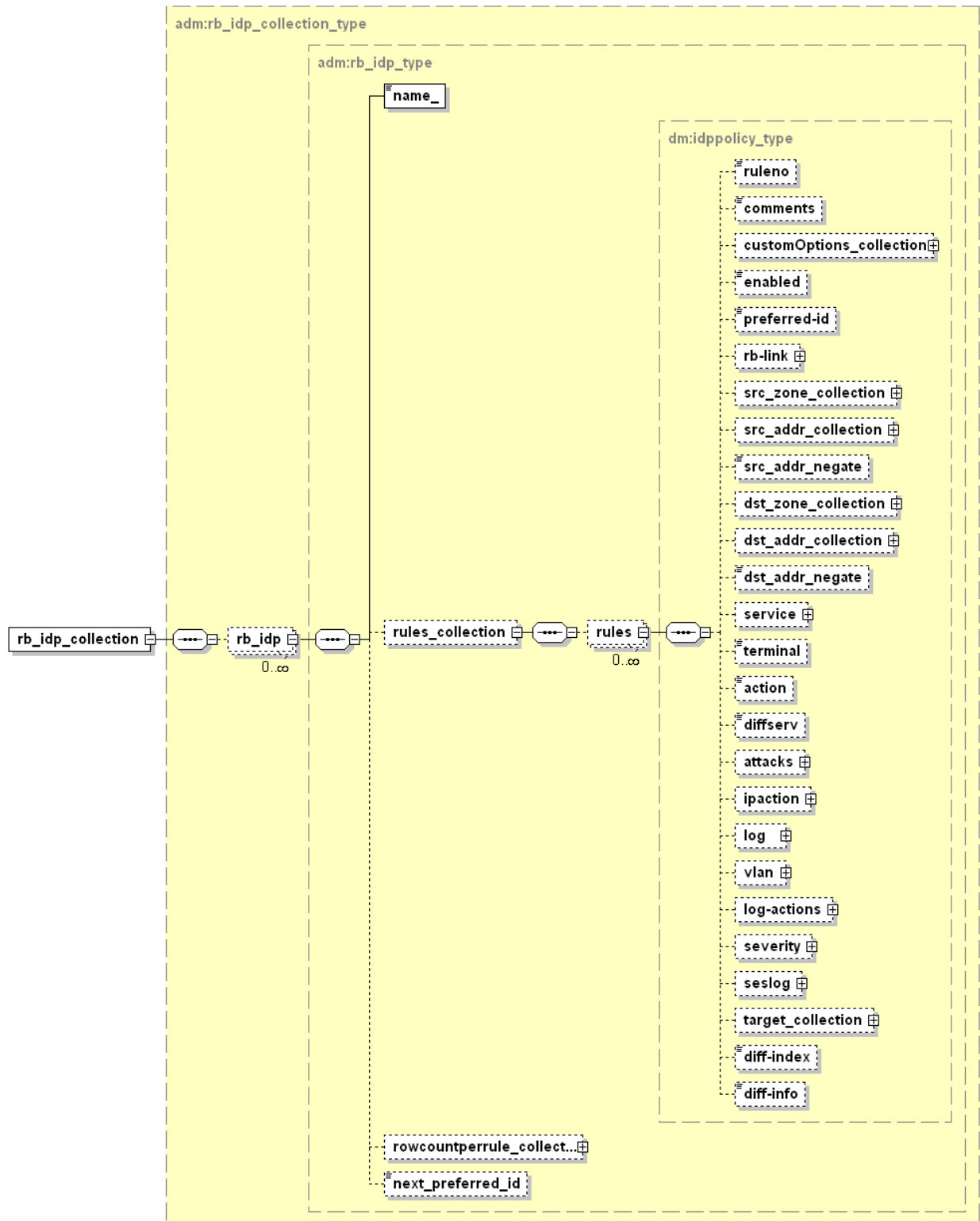


Table 15: IDP Rulebase Data Elements

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.
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.

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

Data Element	Description
action	<p>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:</p> <ul style="list-style-type: none"> <li>• Accept = IDP accepts the interactive traffic</li> <li>• 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.</li> <li>• Close Client = IDP closes the interactive connection to the client but not to the server.</li> <li>• Close Server = IDP closes the interactive connection to the server but not to the client.</li> <li>• 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.</li> </ul>
diffserv	DiffServ Marking.
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	<p>This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> </ul>
log-actions	Action to be taken on the log. This can include configuring SNMP, Syslog, CSV, XML, script, and e-mail settings.
severity	<p>Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings:</p> <ul style="list-style-type: none"> <li>• Default</li> <li>• Info</li> <li>• Warning</li> <li>• Minor</li> <li>• Major</li> <li>• Critical</li> </ul>
seslog	Log packets.

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

Data Element	Description
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 44 and Table 16 on page 44.

Figure 11: Multicast Rulebase

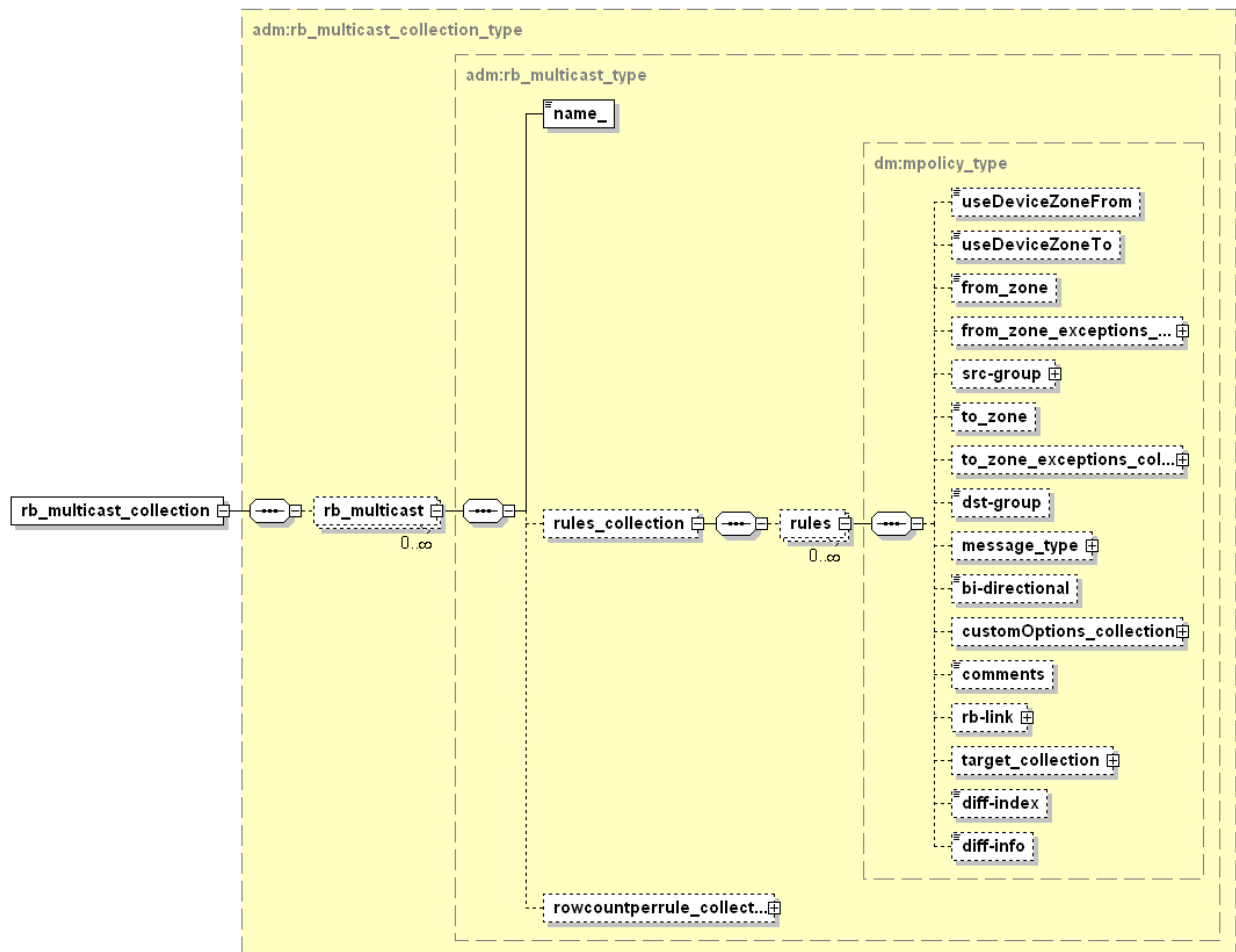


Table 16: Multicast Rulebase Data Elements

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.
from_zone	You must select a single zone for the source zone. The source will send multicast traffic from this zone.

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

Data Element	Description
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 46 and Table 17 on page 46.

Figure 12: SYN Protector Rulebase

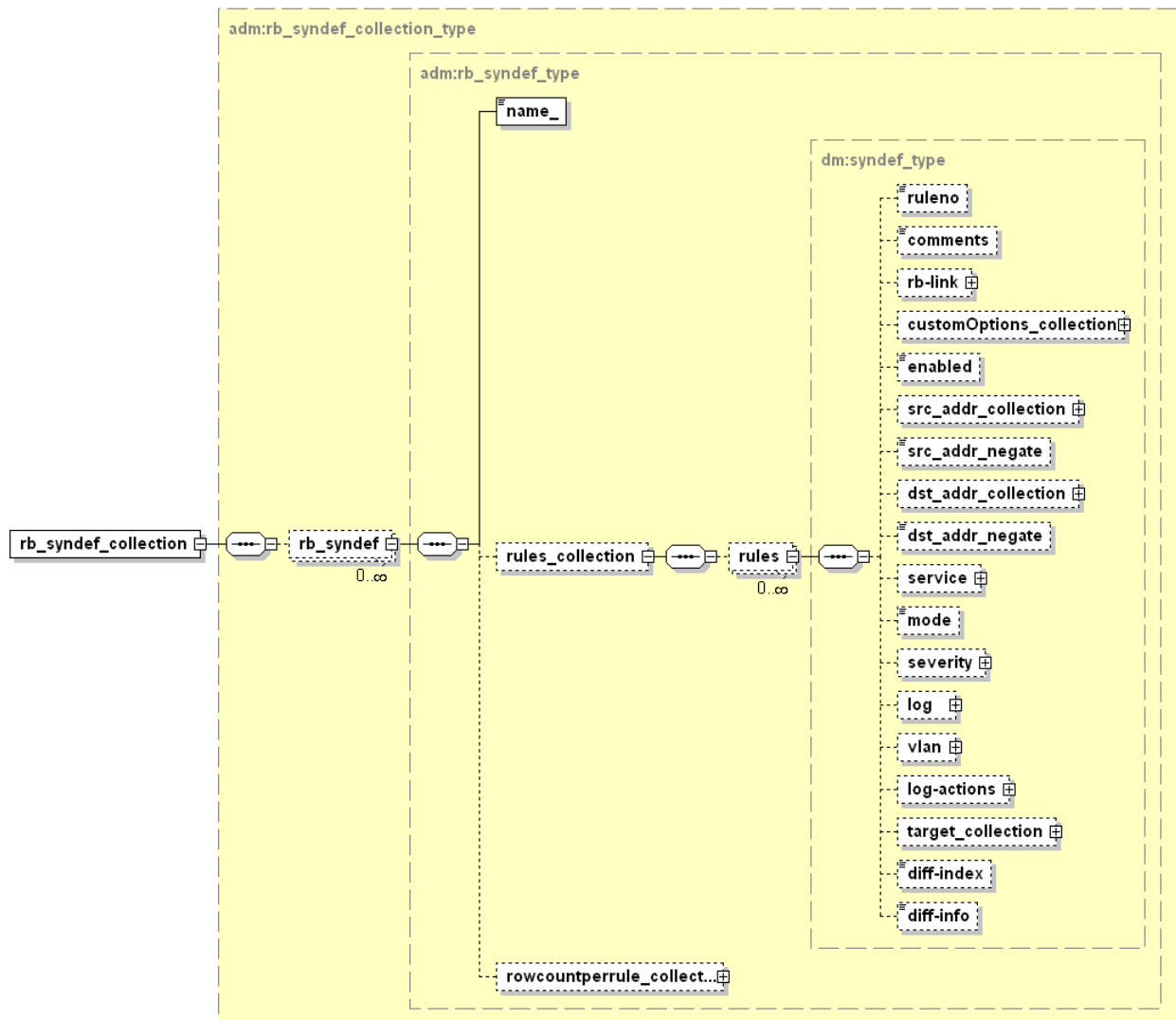


Table 17: SYN Protector Rulebase Data Elements

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.

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

Data Element	Description
comments	Comments about the SYN Protector collection.
customOptions_collection	Custom options.
enabled	Collection enabled.
src_addr_collection	Traffic source address.
src_addr_negate	Negates the specified source address.
dst_addr_coillection	Traffic destination address.
dst_addr_negate	Negates the specified destination address.
service	<p>The default service, <b>TCP-any</b>, looks for SYN floods in all TCP-based traffic.</p> <p><b>NOTE:</b> Always set the SYN Protector service value to <b>TCP-any</b>. Selecting individual services can cause unpredictable interactions with other rulebases.</p>
mode	<p>Select the mode that indicates how IDP handles TCP traffic. The possible values are:</p> <ul style="list-style-type: none"> <li>• None = no action taken.</li> <li>• Relay = IDP acts as the middleman or relay for the established connection.</li> <li>• Passive = IDP handles the transfer of packets between the client host and the server but does not prevent the connection from being established.</li> </ul>
severity	<p>Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings:</p> <ul style="list-style-type: none"> <li>• Default</li> <li>• Info</li> <li>• Warning</li> <li>• Minor</li> <li>• Major</li> <li>• Critical</li> </ul>
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	<p>This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are:</p> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> </ul>

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

Data Element	Description
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 49 and Table 18 on page 49.

Figure 13: Traffic Anomalies Rulebase

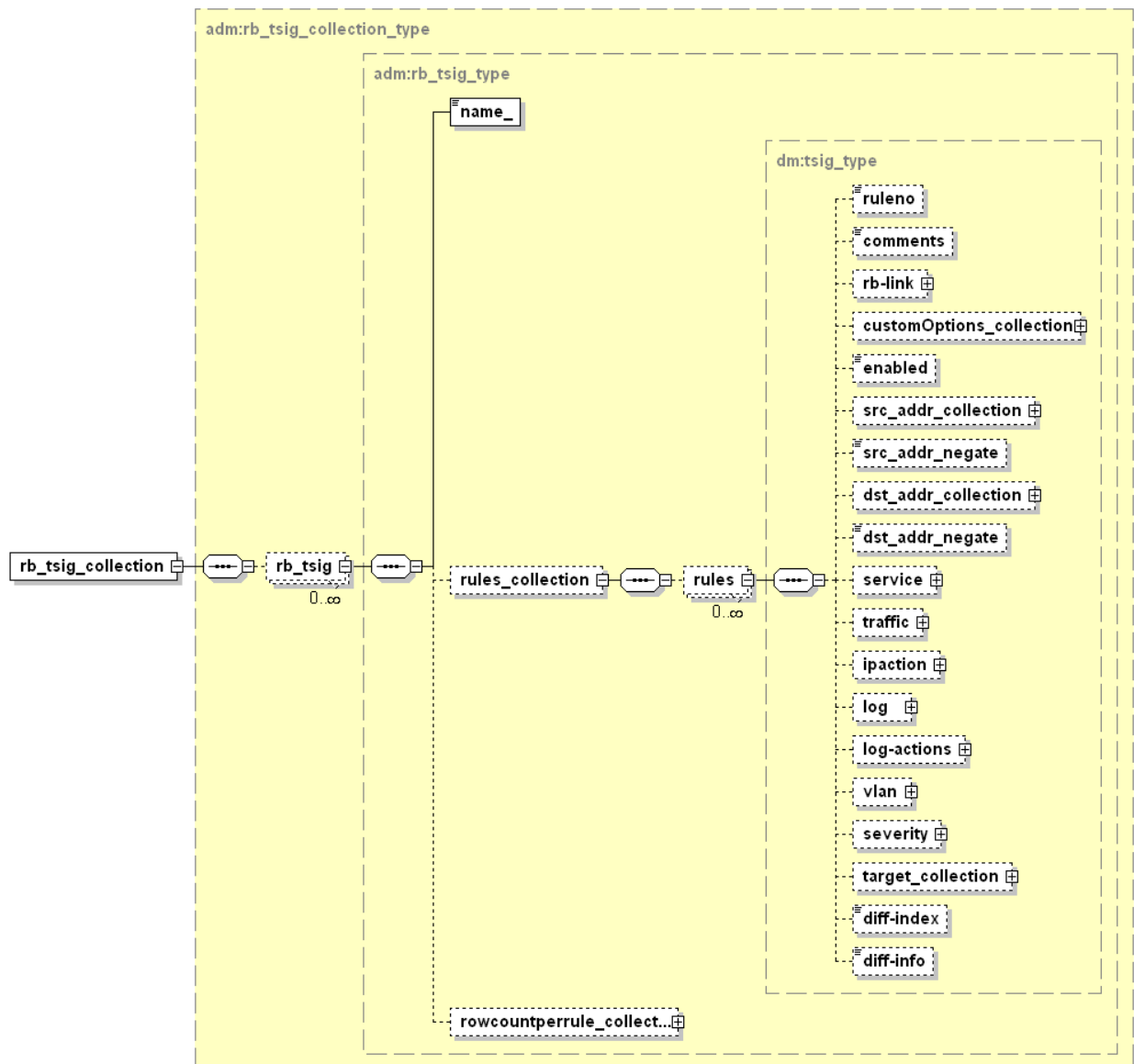


Table 18: Traffic Anomalies Rulebase Data Elements

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 Anomalies Rulebase Data 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: <ul style="list-style-type: none"> <li>Ignore = The IDP Sensor ignores the traffic.</li> <li>Detect = The IDP Sensor detects the traffic but does not log it.</li> <li>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.</li> <li>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.</li> <li>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.</li> <li>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.</li> </ul>
ipaction	Enables and configures an IP action to prevent future malicious connections from the attacker's IP address.
log	GTP logging.

Table 18: Traffic Anomalies Rulebase Data Elements (*continued*)

Data Element	Description
log-actions	Log action settings. Possible settings include configuring: <ul style="list-style-type: none"> <li>• SNMP</li> <li>• Syslog</li> <li>• CVS</li> <li>• XML</li> <li>• script</li> <li>• e-mail</li> </ul>
vlan	This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are: <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> <li>• 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.</li> </ul>
severity	Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings: <ul style="list-style-type: none"> <li>• Default</li> <li>• Info</li> <li>• Warning</li> <li>• Minor</li> <li>• Major</li> <li>• Critical</li> </ul>
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 52 and Table 19 on page 52.

Figure 14: Network Honeypot Rulebase

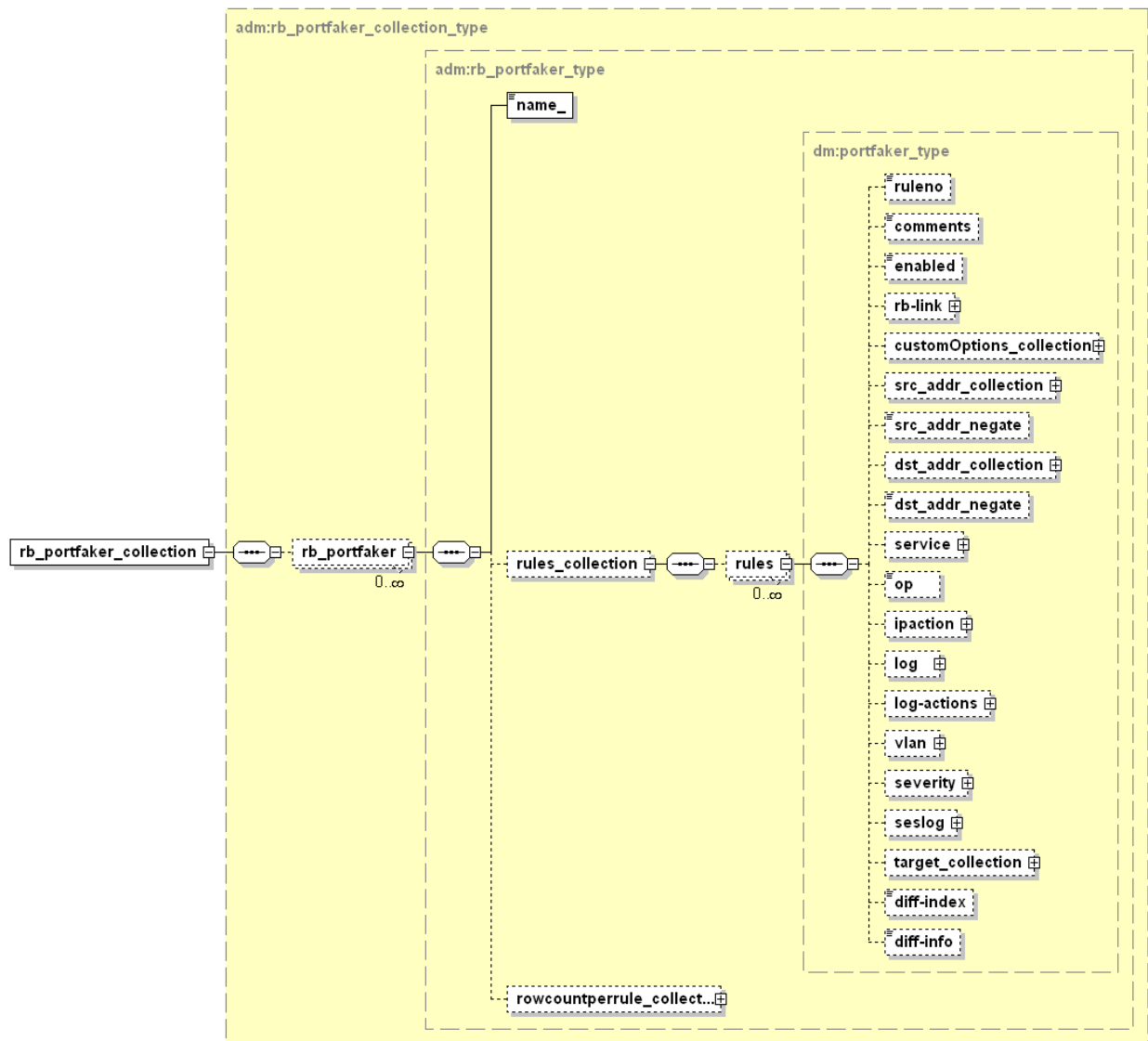


Table 19: Network Honeypot Rulebase Data Elements

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.

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

Data Element	Description
ruleno	Rule number
comments	Comments about the network honeypot (portfaker) collection.
enabled	Collection enabled.
rb_link	Portfaker Link collection
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	<p>This parameter configures a rule that only applies to messages in specified VLANs. The possible settings are:</p> <ul style="list-style-type: none"> <li>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.</li> <li>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.</li> <li>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.</li> </ul>

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

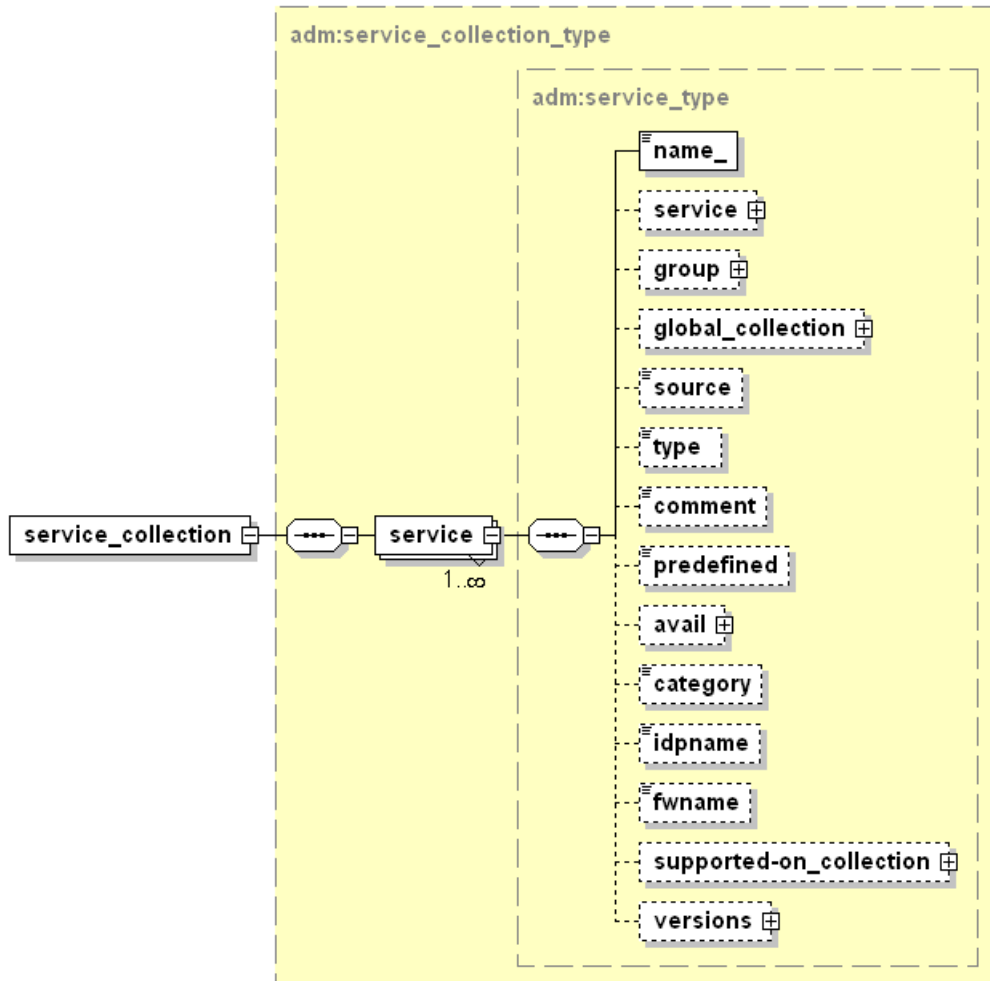
Data Element	Description
severity	Severity of the attack. Within the IDP rulebase, you can override the ordinary attack severity on a per-rule basis. Possible settings: <ul style="list-style-type: none"><li>• Default</li><li>• Info</li><li>• Warning</li><li>• Minor</li><li>• Major</li><li>• Critical</li></ul>
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 55 and Table 20 on page 55.

Figure 15: Service Collection



**NOTE:** Addresses must be created before you can configure a security policy. See “Address (address\_collection\_type)” on page 56.

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.

Table 20: Service Collection Data Elements (*continued*)

Data Element	Description
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 57 and Table 21 on page 57.

Figure 16: Address Collection

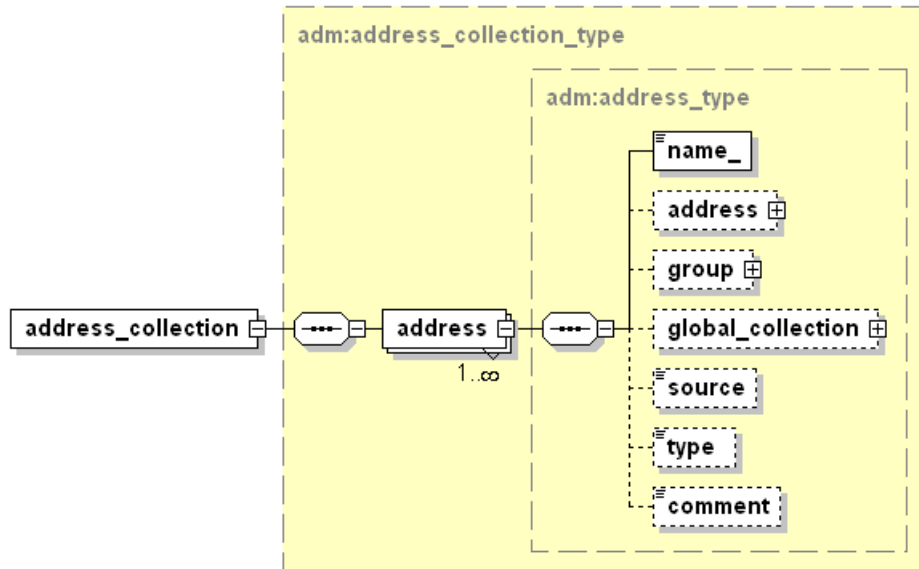


Table 21: Address Collection Data Elements

Data Element	Description
<code>name_</code>	Name of the device or network component.
<code>group</code>	Name of the group.
<code>global_collection</code>	Global rules collection.
<code>source</code>	Source of the address type.
<code>type</code>	Address type (host, network, or group).
<code>comment</code>	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 58 and Table 22 on page 58.

Figure 17: Schedule Object

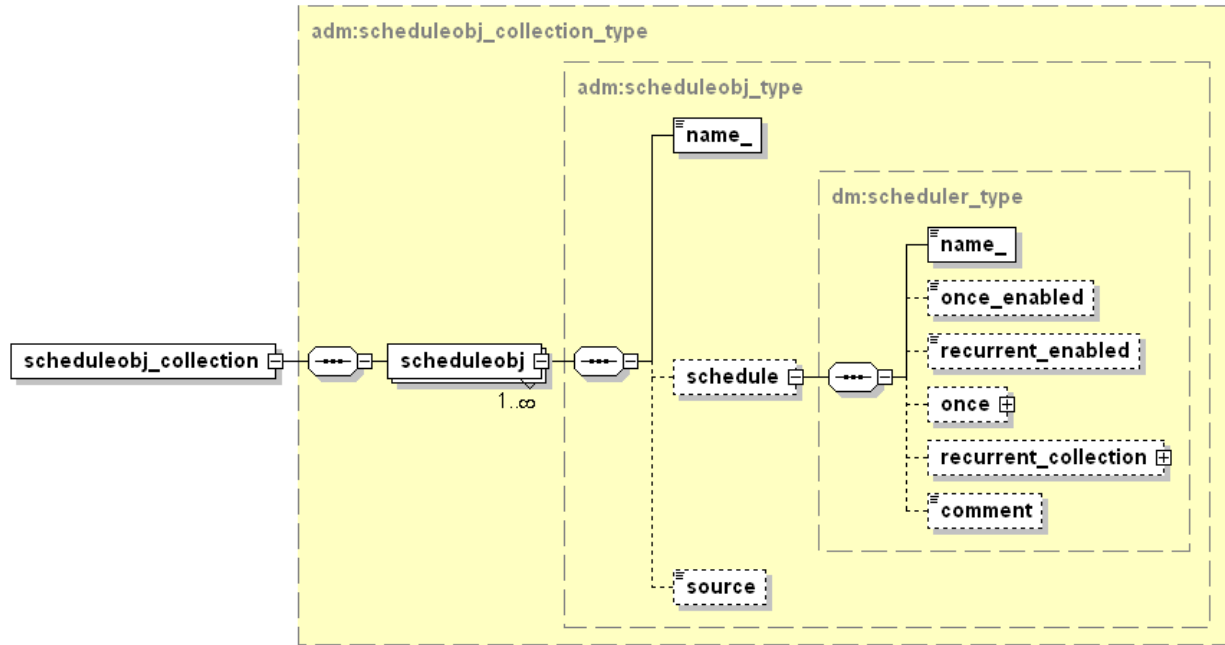


Table 22: Schedule Object Data Elements

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 61.

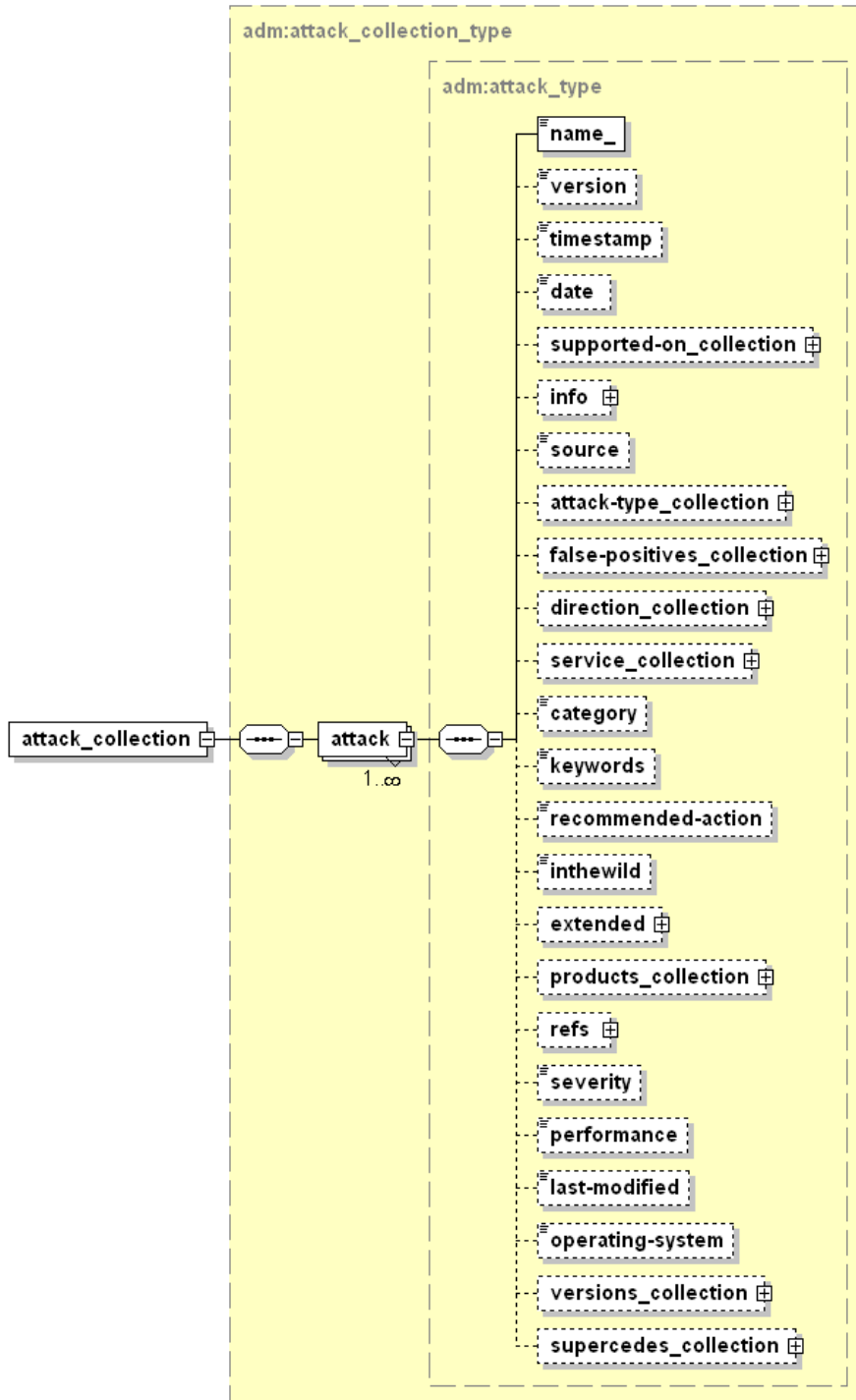


Table 23: Attack Collection Data Elements

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 54.
category	Attack category.
keywords	Keywords associated with the attack.
recommended-action	Recommended action in response to the specified type of attack. Possible values: <ul style="list-style-type: none"> <li>• none (default)</li> <li>• ignore</li> <li>• drop-packet</li> <li>• drop</li> <li>• close-client</li> <li>• close-server</li> <li>• close</li> </ul>
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: <ul style="list-style-type: none"><li>• Critical</li><li>• Major</li><li>• Minor</li><li>• Warning</li><li>• Info</li></ul>
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 63 and Table 24 on page 63.

Figure 18: Antivirus Collection

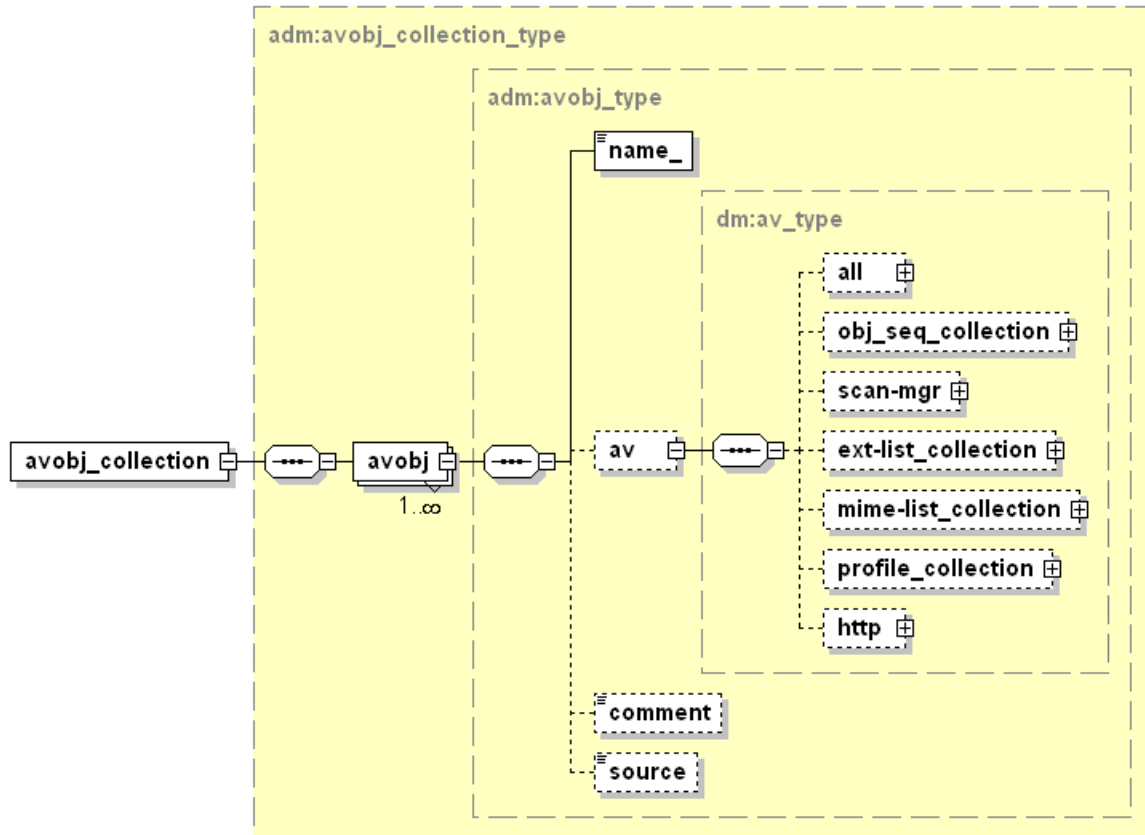


Table 24: Antivirus Collection Data Elements

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

**Table 24: Antivirus Collection Data Elements (continued)**

Data Element	Description
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 65 and Table 25 on page 65.

Figure 19: GTP Collection

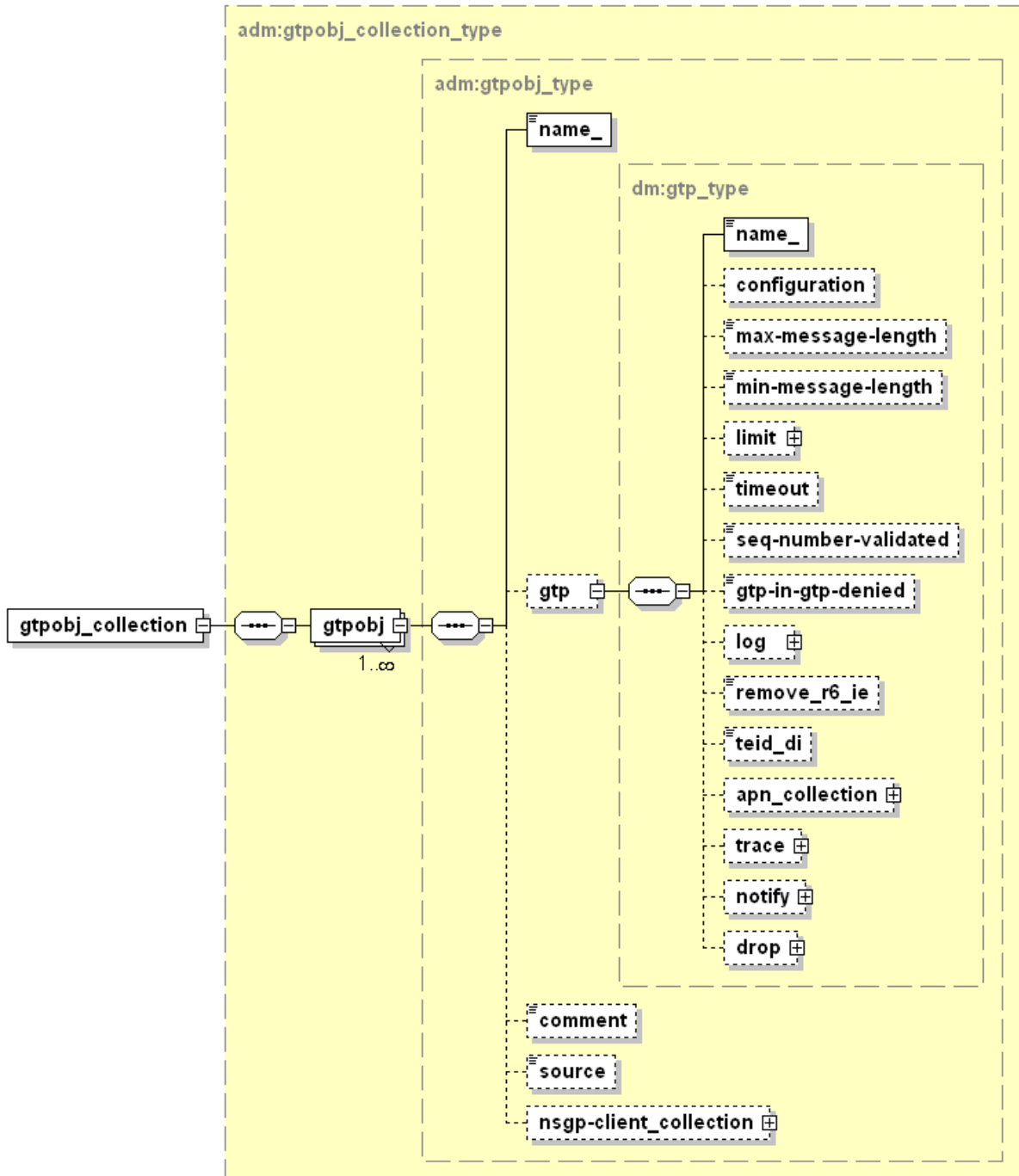


Table 25: GTP Collection Data Elements

Data Element	Description
gtpobj	GTP object

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

Data Element	Description
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.
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: <ul style="list-style-type: none"> <li>• never = no timeout</li> <li>• default = default period of time</li> <li>• user-defined = user defined inactivity timeout period in minutes.</li> </ul>
seq-number-validated	Sequence number validation
gtp-in-gtp-denied	GTP in GTP denied
log	GTP logging.
remove_r6_je	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 67 and Table 26 on page 67.

Figure 20: DI Profile

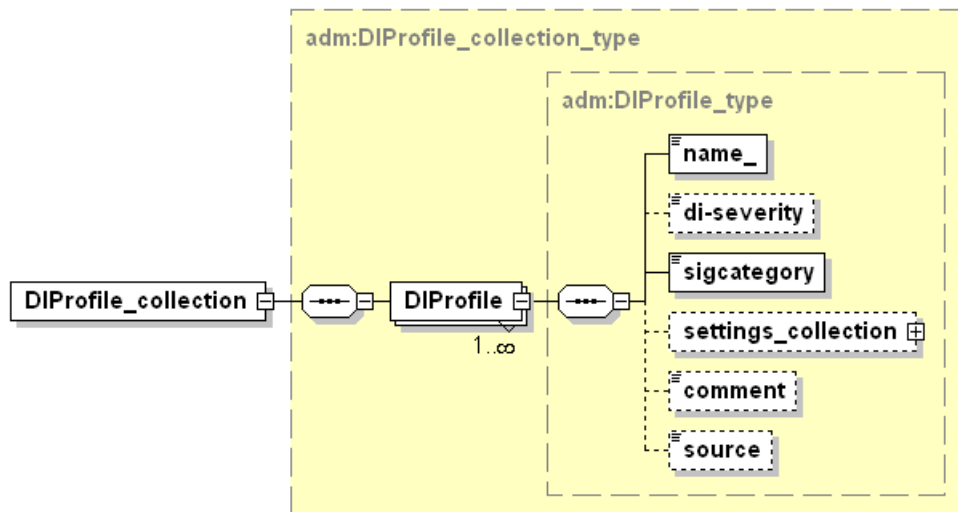


Table 26: DIP Data Elements

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 68 and Table 27 on page 68.

Figure 21: Global DIP Collection

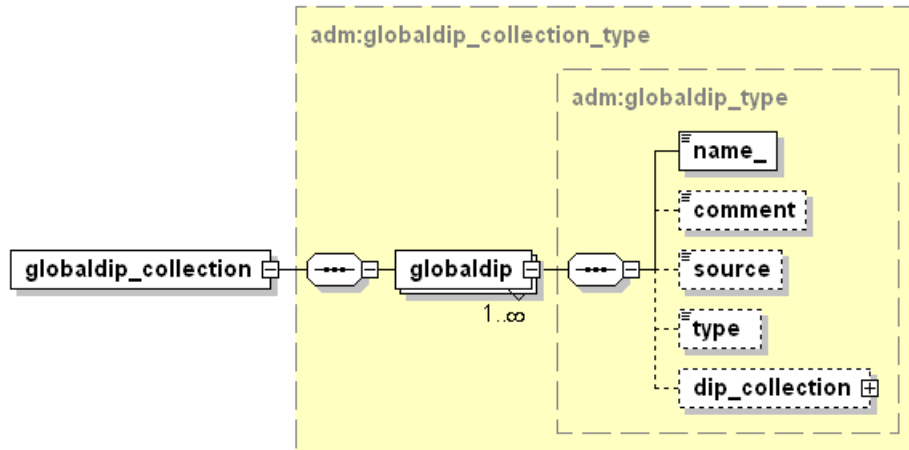


Table 27: Global DIP Data Elements

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 (globalmpi\_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 69 and Table 28 on page 69.

Figure 22: Global MIP Collection

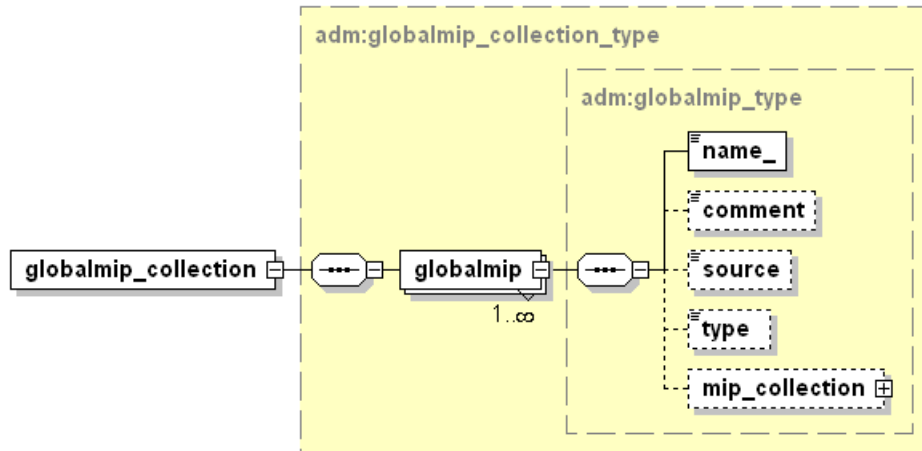


Table 28: Global MIP Data Elements

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 70 and Table 29 on page 70.



Figure 24: URL Filter Object Collection

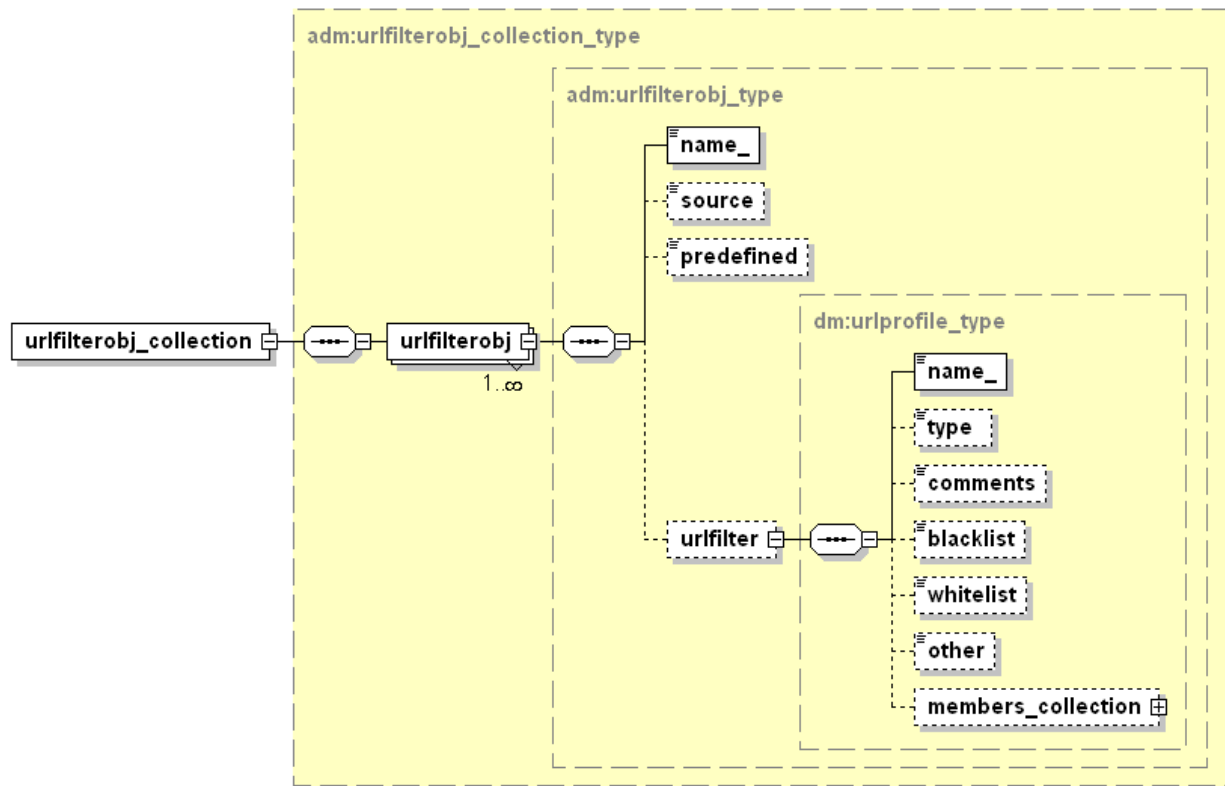


Table 30: URL Filter Data Collection

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)

Table 30: URL Filter Data Collection (*continued*)

Data Element	Description
other	Action for all other URLs.
members_collection	Members categories.

## PART 3

# Using the NSM API from a Perl Client

These chapters explain 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 75
- Using the Perl Client to Access the NSM API on page 81
- Using the API to Manage Shared Objects on page 83



## 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 75
- Installing the Perl Client Environment on Windows Machines on page 76
- Using a Perl Script to Access the NSM API on page 77

## Installing the Perl Client Environment on Linux-Unix Machines

---

Before you can install this client environment, the following software must be installed on your machine:

- Perl version 5.8.8

Perl is available (free) at

<http://www.activestate.com/Products/activeperl/index.mhtml>.

- 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 SOAP Lite (do not accept the default): **cpan[1]> install SOAP::Lite**
10. Enable https, MIME, DIME, and Axis2 MIME support.

## Installing the Perl Client Environment on Windows Machines

---

Before you can install this client environment, the following software must be installed on your machine:

- Perl version 5.8.8

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

- 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 SOAP Lite (do not accept the default): **cpan[1]> install SOAP::Lite**
11. Enable https, MIME, DIME, and Axis2 MIME support.
12. Upgrade all modules, accepting the defaults: **cpan> upgrade**  
The upgrade process takes a few minutes.

## Using a Perl Script to Access the NSM API

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

```
#!/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 $jpr_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/$service")
        -> uri ( "$jpr_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' => "$jpr_url/" .
        lc($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/client` on NSM server.

- Login and Logout on page 81

### Login and Logout

---

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

# Login:

```
my $host = [your hostname or IP here]
my $connect = MAIN::NSM->new('HOST'=>"$host");
$connect->login;
```

#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 83
- Using the Perl Client Library with Service Objects on page 87
- Using the Perl Client Library with Device Objects on page 91

## 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 into 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.

- Add a host with IP/Mask.

Enter:

```

$result = $address->addHostObjects('OBJECT_NAME'=>'Host_IP-' . $i,
'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-' . $i);
    put_log('WARN', msg=>"Error
while adding Host objects." ) unless( $result );

$result = $address->addHostObjects('OBJECT_NAME'=>'Foo-' . $i,
'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-' . $i,
'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-' . $i,
'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'=>'wilma',
               '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:

```

for( my $i = 1; $i < 10; $i++ ){
                                $result =
$address->replaceHostObjects('OBJECT_NAME'=>'Foo-'. $i,
'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:

```
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'}");
}

( $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"
    );
    }
}

( $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();
        put_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:

```
'IS_ICMP'=>[
                                                    {'TYPE'=>'10',
'CODE'=>'11'},
                                                    {'TYPE'=>'20', 'CODE'=>'21'}
]
};

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

$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 );

$members = {
                                                    'SUN_RPC'=>[
{'SUN_LOW'=>'101'},
                                                    {'SUN_LOW'=>'201'}
]};

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

```

$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:

```

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 );

$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:

```
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" );
        }
    }
}

( $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" );
    }
}

( $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.

## 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 95
- Using APIs for Policy Management on page 97
- Using APIs for Shared Object Management on page 105
- Using APIs for Job Management on page 111
- Using APIs for Device Management on page 117



## 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 95
- Logout on page 96

## 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.getLoginStatusCode().
    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();
    }
}
```

# 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 97
- Update an Existing Policy on page 99
- Delete a Policy on page 100
- Get a List of Policies on page 101
- Get a Policy on page 101
- Assign a Policy to a Device on page 102
- Remove a Policy Assignment on page 103

## 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>
```

```
testRuleBaseFirewallInsert.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("./nsmppolicy-id"):
        pathValueType.setValue(this.createOpaqueDataType
           ("<nsmppolicy-id>&l.nsmppolicy.????????test </nsmppolicy-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();
}
}
```

# 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 105
- Replace a Shared Object on page 106
- Delete a Shared Object on page 108
- Get a List of Shared Objects on page 108
- Get a Shared Object on page 109

## 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>
    <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>
</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();
    }
}

```



## Using APIs for Job Management

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

- Get a Job Result on page 111
- Import a List of Devices on page 112
- Update a List of Devices on page 113
- Get a Configuration Summary on page 113
- Get a Running Configuration on page 114
- Get the Delta Configuration on page 115
- Cancel a Job Request on page 115

### 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();
    }
}
```

# Using APIs for Device Management

- Retrieve Domains on page 117
- Retrieve the Device List in One Domain on page 117

## 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 address 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();
    }
}
```

## 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 121
- System Service API WSDL on page 129
- Data Centric API WSDL on page 135
- Log Service API WSDL on page 145



# Job Service API WSDL

This chapter describes the Job Service API WSDL.

- WSDL File on page 121

## 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: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="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>
<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="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="GetDeltaConfigRequest" 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="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="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>
```

# System Service API WSDL

This chapter describes the System Service API WSDL.

- WSDL File on page 129

## 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:sequence>

```

```

        <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>
```



# Data Centric API WSDL

This chapter describes the Data Centric API WSDL.

- WSDL File on page 135

## 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="objectData" 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="objectData" type="core:ObjectDataType"/>
    <element name="objectIdentifier" type="core:ObjectIdentifierType"/>
  </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="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"/>
  </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="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="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>

```

```

        </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 repository

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:NameValuePair" 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="GetObjectViewByIdRequest"
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>
  <output>
    <soap:body use="literal"/>
  </output>
</wsdl:operation>
<wsdl:operation name="ResolveObjectReferenceRequest">
  <soap:operation soapAction="urn:#ResolveObjectReferenceRequest"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</wsdl:operation>
<wsdl:operation name="QueryObjectViewRequest">
  <soap:operation soapAction="urn:#QueryObjectViewRequest"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <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>

```

```
</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>
```

# Log Service API WSDL

This chapter describes the Log Service API WSDL.

- WSDL File on page 145

## 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"/>
              </sequence>
            </extension>
          </complexContent>
        </complexType>
      </element>
    </xs:schema>
  </wsdl:types>
</wsdl:definitions>
```

```

                <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 149



# 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.....	29
backdoor.....	25
critical.....	29
IDP will exempt.....	30
information.....	29
objects.....	42
service used in.....	28
severity of IDP rulebase.....	30
tag.....	29

authentication.....	4, 95
---------------------	-------

authorization.....	4
--------------------	---

## B

### backdoor\_type

action.....	29
comments.....	28
customOptions_collection.....	28
dst_addr_collection.....	28
dst_addr_negate.....	28
dst_zone_collection.....	28
enabled.....	28
log.....	29
log-actions.....	29
op.....	29
preferred-id.....	28
rb-link.....	28
rueno.....	28
service.....	28
seslog.....	30
severity.....	30
src_addr_collection.....	28

src_addr_negate.....	28
----------------------	----

src_zone_collection.....	28
--------------------------	----

target_collection.....	30
------------------------	----

vlan.....	29
-----------	----

## C

CommonDataTypes.xsd.....	4
--------------------------	---

customer support.....	xvii
-----------------------	------

contacting JTAC.....	xvii
----------------------	------

## D

Data Centric API.....	97
-----------------------	----

GetObjectDependentRequest.....	9
--------------------------------	---

GetObjectViewByCategoryRequest.....	9
-------------------------------------	---

GetObjectViewByIdRequest.....	10
-------------------------------	----

LockObjectRequest.....	10
------------------------	----

ModifyObjectViewRequest.....	10
------------------------------	----

QueryObjectViewRequest.....	11
-----------------------------	----

ResolveObjectReferenceRequest.....	11
------------------------------------	----

UnlockObjectRequest.....	10
--------------------------	----

using for device management.....	8, 117
----------------------------------	--------

using for policy management.....	8
----------------------------------	---

using for shared object management.....	8
---	---

using to manage shared objects.....	105
-------------------------------------	-----

### 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.....	xv, 11
---------------------------	--------

**E**

error handling.....4

**F**

## firewall policy type

- action.....35
- anti-spam.....39
- application.....38
- application-services.....39
- attack-policy.....38
- auths.....37
- av.....39
- count.....36
- grp.....39
- ha\_session\_backup.....37
- idp.....38
- log.....36
- log-actions.....36
- name\_.....35
- nat.....36
- nat-dip.....36
- no-idle-reset.....37
- pol\_name.....35
- preferred-id.....36
- schedule.....37
- service\_collection.....35
- target\_collection.....36
- traffic.....37
- url-blk.....36
- url-protocol.....36

**I**

## IDP

- attacks exempted.....32
- detects interactive traffic.....25
- rulebase.....25, 30, 51
- rules.....39

IDP Sensor.....50

*See also* IDP

## idppolicy\_type

- action.....42
- attacks.....42
- comments.....41
- customOptions\_collection.....41
- diffserv.....42
- dst\_addr\_collection.....41
- dst\_addr\_negate.....41
- dst\_zone\_collection.....41
- enabled.....41

- ipaction.....42
- log.....42
- log-actions.....42
- preferred-id.....41
- rb-link.....41
- rueno.....41
- service.....41
- seslog.....42
- severity.....42
- src\_addr\_collection.....41
- src\_addr\_negate.....41
- src\_zone\_collection.....41
- target\_collection.....43
- terminal.....41
- vlan.....42

## 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, 111

**L**

## log

- action settings.....51
- alert.....42
- attack.....47
- audit.....20
- data and packet data.....14
- event.....14
- GTP.....50
- packet.....42
- packet data retrieval.....4
- packets.....30
- record.....30

## 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).....	36
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.....	23
See also policy_type	

**P**

policy_type	
accesstype.....	23
comment.....	23
createFrom.....	23
name_.....	23
rulebases.....	23
port scan	
distributed.....	50
TCP.....	50
UDP.....	50

**R**

rb_backdoor_type	
name_.....	27
rules.....	27
rules_collection.....	27
rb_idp_type	
name_.....	41
next_preferred_id.....	41
rowcountperrule_collection.....	41
rules_collection.....	41
rule	
attach matching.....	29
backdoor.....	26
global.....	25
row count per.....	41
rule ID.....	28, 32, 36, 41
service object.....	28

## rulebases

backdoor.....	25
exempt.....	30
firewall.....	33
IDP.....	39
multicast.....	43
rules.....	25
SYN protector.....	45

**S**

security policy	
rulebases.....	25
SOAP/HTTPS.....	3
subtree filter	
attribute matching expressions.....	8
containment nodes.....	8
content matching nodes.....	8
namespace selection.....	8
selection nodes.....	8
support, technical See technical support	
System Service API.....	95
GetSystemInfoRequest.....	8
LoginRequest.....	7
LogoutRequest.....	8
RespondToChallengeRequest.....	8
using for authentication.....	7
using for authorization.....	7

**T**

technical support	
contacting JTAC.....	xvii
third-party developers.....	3
traffic	
flow.....	25
matching to rules.....	25

**V**

vlan	
messages in specified.....	29
tag.....	51
VLAN tag.....	32

**W**

Web Service Definition Language.....	3
WSDL.....	3
Data Centric API.....	135
Job Service API.....	121
Log Service API.....	145
System Service API.....	129

X

XML

- schema.....3
- subtree filter.....8
  - See also subtree filter

XSD.....3

- definition files.....23