



Alcatel-Lucent 5620

SERVICE AWARE MANAGER | RELEASE 9.0 R7
XML OSS INTERFACE DEVELOPER GUIDE

3HE 06498 AAAG TQZZA Edition 01

Alcatel-Lucent assumes no responsibility for the accuracy of the information presented, which is subject to change without notice.

Alcatel, Lucent, Alcatel-Lucent, the Alcatel-Lucent logo, and TiMetra are registered trademarks of Alcatel-Lucent. All other trademarks are the property of their respective owners.

Copyright 2011-2012 Alcatel-Lucent.
All rights reserved.

Disclaimers

Alcatel-Lucent products are intended for commercial uses. Without the appropriate network design engineering, they must not be sold, licensed or otherwise distributed for use in any hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life-support machines, or weapons systems, in which the failure of products could lead directly to death, personal injury, or severe physical or environmental damage. The customer hereby agrees that the use, sale, license or other distribution of the products for any such application without the prior written consent of Alcatel-Lucent, shall be at the customer's sole risk. The customer hereby agrees to defend and hold Alcatel-Lucent harmless from any claims for loss, cost, damage, expense or liability that may arise out of or in connection with the use, sale, license or other distribution of the products in such applications.

This document may contain information regarding the use and installation of non-Alcatel-Lucent products. Please note that this information is provided as a courtesy to assist you. While Alcatel-Lucent tries to ensure that this information accurately reflects information provided by the supplier, please refer to the materials provided with any non-Alcatel-Lucent product and contact the supplier for confirmation. Alcatel-Lucent assumes no responsibility or liability for incorrect or incomplete information provided about non-Alcatel-Lucent products.

However, this does not constitute a representation or warranty. The warranties provided for Alcatel-Lucent products, if any, are set forth in contractual documentation entered into by Alcatel-Lucent and its customers.

This document was originally written in English. If there is any conflict or inconsistency between the English version and any other version of a document, the English version shall prevail.

Alcatel-Lucent License Agreement

SAMPLE END USER LICENSE AGREEMENT

1. LICENSE

- 1.1 Subject to the terms and conditions of this Agreement, Alcatel-Lucent grants to Customer and Customer accepts a nonexclusive, nontransferable license to use any software and related documentation provided by Alcatel-Lucent pursuant to this Agreement ("Licensed Program") for Customer's own internal use, solely in conjunction with hardware supplied or approved by Alcatel-Lucent. In case of equipment failure, Customer may use the Licensed Program on a backup system, but only for such limited time as is required to rectify the failure.
- 1.2 Customer acknowledges that Alcatel-Lucent may have encoded within the Licensed Program optional functionality and capacity (including, but not limited to, the number of equivalent nodes, delegate workstations, paths and partitions), which may be increased upon the purchase of the applicable license extensions.
- 1.3 Use of the Licensed Program may be subject to the issuance of an application key, which shall be conveyed to the Customer in the form of a Supplement to this End User License Agreement. The purchase of a license extension may require the issuance of a new application key.

2. PROTECTION AND SECURITY OF LICENSED PROGRAMS

- 2.1 Customer acknowledges and agrees that the Licensed Program contains proprietary and confidential information of Alcatel-Lucent and its third party suppliers, and agrees to keep such information confidential. Customer shall not disclose the Licensed Program except to its employees having a need to know, and only after they have been advised of its confidential and proprietary nature and have agreed to protect same.
- 2.2 All rights, title and interest in and to the Licensed Program, other than those expressly granted to Customer herein, shall remain vested in Alcatel-Lucent or its third party suppliers. Customer shall not, and shall prevent others from copying, translating, modifying, creating derivative works, reverse engineering, decompiling, encumbering or otherwise using the Licensed Program except as specifically authorized under this Agreement. Notwithstanding the foregoing, Customer is authorized to make one copy for its archival purposes only. All appropriate copyright and other proprietary notices and legends shall be placed on all Licensed Programs supplied by Alcatel-Lucent, and Customer shall maintain and reproduce such notices on any full or partial copies made by it.

3. TERM

- 3.1 This Agreement shall become effective for each Licensed Program upon delivery of the Licensed Program to Customer.

-
- 3.2 Alcatel-Lucent may terminate this Agreement: (a) upon notice to Customer if any amount payable to Alcatel-Lucent is not paid within thirty (30) days of the date on which payment is due; (b) if Customer becomes bankrupt, makes an assignment for the benefit of its creditors, or if its assets vest or become subject to the rights of any trustee, receiver or other administrator; (c) if bankruptcy, reorganization or insolvency proceedings are instituted against Customer and not dismissed within 15 days; or (d) if Customer breaches a material provision of this Agreement and such breach is not rectified within 15 days of receipt of notice of the breach from Alcatel-Lucent.
- 3.3 Upon termination of this Agreement, Customer shall return or destroy all copies of the Licensed Program. All obligations of Customer arising prior to termination, and those obligations relating to confidentiality and nonuse, shall survive termination.

4. CHARGES

- 4.1 Upon shipment of the Licensed Program, Alcatel-Lucent will invoice Customer for all fees, and any taxes, duties and other charges. Customer will be invoiced for any license extensions upon delivery of the new software application key or, if a new application key is not required, upon delivery of the extension. All amounts shall be due and payable within thirty (30) days of receipt of invoice, and interest will be charged on any overdue amounts at the rate of 1 1/2% per month (19.6% per annum).

5. SUPPORT AND UPGRADES

- 5.1 Customer shall receive software support and upgrades for the Licensed Program only to the extent provided for in the applicable Alcatel-Lucent software support policy in effect from time to time, and upon payment of any applicable fees. Unless expressly excluded, this Agreement shall be deemed to apply to all updates, upgrades, revisions, enhancements and other software which may be supplied by Alcatel-Lucent to Customer from time to time.

6. WARRANTIES AND INDEMNIFICATION

- 6.1 Alcatel-Lucent warrants that the Licensed Program as originally delivered to Customer will function substantially in accordance with the functional description set out in the associated user documentation for a period of 90 days from the date of shipment, when used in accordance with the user documentation. Alcatel-Lucent's sole liability and Customer's sole remedy for a breach of this warranty shall be Alcatel-Lucent's good faith efforts to rectify the nonconformity or, if after repeated efforts Alcatel-Lucent is unable to rectify the nonconformity, Alcatel-Lucent shall accept return of the Licensed Program and shall refund to Customer all amounts paid in respect thereof. This warranty is available only once in respect of each Licensed Program, and is not renewed by the payment of an extension charge or upgrade fee.

-
- 6.2 ALCATEL-LUCENT EXPRESSLY DISCLAIMS ALL OTHER WARRANTIES, REPRESENTATIONS, COVENANTS OR CONDITIONS OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, WARRANTIES OR REPRESENTATIONS OF WORKMANSHIP, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, DURABILITY, OR THAT THE OPERATION OF THE LICENSED PROGRAM WILL BE ERROR FREE OR THAT THE LICENSED PROGRAMS WILL NOT INFRINGE UPON ANY THIRD PARTY RIGHTS.
- 6.3 Alcatel-Lucent shall defend and indemnify Customer in any action to the extent that it is based on a claim that the Licensed Program furnished by Alcatel-Lucent infringes any patent, copyright, trade secret or other intellectual property right, provided that Customer notifies Alcatel-Lucent within ten (10) days of the existence of the claim, gives Alcatel-Lucent sole control of the litigation or settlement of the claim, and provides all such assistance as Alcatel-Lucent may reasonably require. Notwithstanding the foregoing, Alcatel-Lucent shall have no liability if the claim results from any modification or unauthorized use of the Licensed Program by Customer, and Customer shall defend and indemnify Alcatel-Lucent against any such claim.
- 6.4 Alcatel-Lucent Products are intended for standard commercial uses. Without the appropriate network design engineering, they must not be sold, licensed or otherwise distributed for use in any hazardous environments requiring fail safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life-support machines, or weapons systems, in which the failure of products could lead directly to death, personal injury, or severe physical or environmental damage. The Customer hereby agrees that the use, sale, license or other distribution of the Products for any such application without the prior written consent of Alcatel-Lucent, shall be at the Customer's sole risk. The Customer also agrees to defend and hold Alcatel-Lucent harmless from any claims for loss, cost, damage, expense or liability that may arise out of or in connection with the use, sale, license or other distribution of the Products in such applications.

7. LIMITATION OF LIABILITY

- 7.1 IN NO EVENT SHALL THE TOTAL COLLECTIVE LIABILITY OF ALCATEL-LUCENT, ITS EMPLOYEES, DIRECTORS, OFFICERS OR AGENTS FOR ANY CLAIM, REGARDLESS OF VALUE OR NATURE, EXCEED THE AMOUNT PAID UNDER THIS AGREEMENT FOR THE LICENSED PROGRAM THAT IS THE SUBJECT MATTER OF THE CLAIM. IN NO EVENT SHALL THE TOTAL COLLECTIVE LIABILITY OF ALCATEL-LUCENT, ITS EMPLOYEES, DIRECTORS, OFFICERS OR AGENTS FOR ALL CLAIMS EXCEED THE TOTAL AMOUNT PAID BY CUSTOMER TO ALCATEL-LUCENT HEREUNDER. NO PARTY SHALL BE LIABLE FOR ANY INDIRECT, SPECIAL OR CONSEQUENTIAL DAMAGES, WHETHER OR NOT SUCH DAMAGES ARE FORESEEABLE, AND/OR THE PARTY HAD BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
- 7.2 The foregoing provision limiting the liability of Alcatel-Lucent's employees, agents, officers and directors shall be deemed to be a trust provision, and shall be enforceable by such employees, agents, officers and directors as trust beneficiaries.

8. GENERAL

- 8.1 Under no circumstances shall either party be liable to the other for any failure to perform its obligations (other than the payment of any monies owing) where such failure results from causes beyond that party's reasonable control.
- 8.2 This Agreement constitutes the entire agreement between Alcatel-Lucent and Customer and supersedes all prior oral and written communications. All amendments shall be in writing and signed by authorized representatives of both parties.
- 8.3 If any provision of this Agreement is held to be invalid, illegal or unenforceable, it shall be severed and the remaining provisions shall continue in full force and effect.
- 8.4 The Licensed Program may contain freeware or shareware obtained by Alcatel-Lucent from a third party source. No license fee has been paid by Alcatel-Lucent for the inclusion of any such freeware or shareware, and no license fee is charged to Customer for its use. The Customer agrees to be bound by any license agreement for such freeware or shareware. CUSTOMER ACKNOWLEDGES AND AGREES THAT THE THIRD PARTY SOURCE PROVIDES NO WARRANTIES AND SHALL HAVE NO LIABILITY WHATSOEVER IN RESPECT OF CUSTOMER'S POSSESSION AND/OR USE OF THE FREWARE OR SHAREWARE.
- 8.5 Alcatel-Lucent shall have the right, at its own expense and upon reasonable written notice to Customer, to periodically inspect Customer's premises and such documents as it may reasonably require, for the exclusive purpose of verifying Customer's compliance with its obligations under this Agreement.
- 8.6 All notices shall be sent to the parties at the addresses listed above, or to any such address as may be specified from time to time. Notices shall be deemed to have been received five days after deposit with a post office when sent by registered or certified mail, postage prepaid and receipt requested.
- 8.7 If the Licensed Program is being acquired by or on behalf of any unit or agency of the United States Government, the following provision shall apply: If the Licensed Program is supplied to the Department of Defense, it shall be classified as "Commercial Computer Software" and the United States Government is acquiring only "restricted rights" in the Licensed Program as defined in DFARS 227-7202-1(a) and 227.7202-3(a), or equivalent. If the Licensed Program is supplied to any other unit or agency of the United States Government, rights will be defined in Clause 52.227-19 or 52.227-14 of the FAR, or if acquired by NASA, Clause 18-52.227-86(d) of the NASA Supplement to the FAR, or equivalent. If the software was acquired under a contract subject to the October 1988 Rights in Technical Data and Computer Software regulations, use, duplication and disclosure by the Government is subject to the restrictions set forth in DFARS 252-227.7013(c)(1)(ii) 1988, or equivalent.
- 8.8 Customer shall comply with all export regulations pertaining to the Licensed Program in effect from time to time. Without limiting the generality of the foregoing, Customer expressly warrants that it will not directly or indirectly export, reexport, or transship the Licensed Program in violation of any export laws, rules or regulations of Canada, the United States or the United Kingdom.

-
- 8.9 No term or provision of this Agreement shall be deemed waived and no breach excused unless such waiver or consent is in writing and signed by the party claimed to have waived or consented. The waiver by either party of any right hereunder, or of the failure to perform or of a breach by the other party, shall not be deemed to be a waiver of any other right hereunder or of any other breach or failure by such other party, whether of a similar nature or otherwise.
- 8.10 This Agreement shall be governed by and construed in accordance with the laws of the Province of Ontario. The application of the United Nations Convention on Contracts for the International Sale of Goods is hereby expressly excluded.

Preface

The Preface provides general information about the 5620 Service Aware Manager documentation suite, including this guide.

Prerequisites

Readers of the 5620 SAM documentation suite are assumed to be familiar with the following:

- 5620 SAM software structure and components
- 5620 SAM GUI operations and tools
- typical 5620 SAM management tasks and procedures
- device and network management concepts

5620 SAM documentation suite

The 5620 SAM documentation suite describes the 5620 SAM and the associated network management of its supported devices. Contact your Alcatel-Lucent support representative for information about specific network or facility considerations.

Table 1 lists the documents in the 5620 SAM customer documentation suite.

Table 1 5620 SAM customer documentation suite

Guide	Description
5620 SAM core documentation	
<i>5620 SAM Release Description</i>	The <i>5620 SAM Release Description</i> provides information about the new features associated with a 5620 SAM software release.

(1 of 4)

Guide	Description
<i>5620 SAM Planning Guide</i>	The <i>5620 SAM Planning Guide</i> provides information about 5620 SAM scalability and recommended hardware configurations.
<i>5620 SAM System Architecture Guide</i>	The <i>5620 SAM System Architecture Guide</i> is intended for technology officers and network planners to increase their knowledge of the 5620 SAM software structure and components. It describes the system structure, software components, and interfaces of the 5620 SAM. In addition, 5620 SAM fault tolerance, security, and network management capabilities are discussed from an architectural perspective.
<i>5620 SAM 5650 CPAM Installation and Upgrade Guide</i>	The <i>5620 SAM 5650 CPAM Installation and Upgrade Guide</i> provides OS considerations, configuration information, and procedures for the following: <ul style="list-style-type: none"> installing, upgrading, and uninstalling 5620 SAM and 5650 CPAM software in standalone and redundant deployments 5620 SAM system migration to a different system conversion from a standalone to a redundant 5620 SAM system
<i>5620 SAM User Guide</i>	The <i>5620 SAM User Guide</i> provides information about using the 5620 SAM to manage the service-aware IP/MPLS network, including GUI basics, commissioning, service configuration, and policy management. The <i>5620 SAM User Guide</i> uses a task-based format. Each chapter contains: <ul style="list-style-type: none"> a workflow that describes the steps for configuring and using the functions detailed procedures that list the configurable parameters on the associated forms 5620 SAM management information specific to LTE network elements is covered in the <i>5620 SAM LTE ePC User Guide</i> and <i>5620 SAM LTE RAN User Guide</i> . 5620 SAM management information specific to 1830 PSS network elements is covered in the <i>5620 SAM Optical User Guide</i> .
<i>5620 SAM Integration Guide</i>	The <i>5620 SAM Integration Guide</i> provides procedures to allow the 5620 SAM to integrate with additional components.
<i>5620 SAM Supervision Module User Guide</i>	The <i>5620 SAM Supervision Module User Guide</i> provides information about how to configure and use the web-based 5620 SAM Supervision Module for fault management and at-a-glance network element monitoring.
<i>5620 SAM Scripts and Templates Developer Guide</i>	The <i>5620 SAM Scripts and Templates Developer Guide</i> provides information that allows you to develop, manage, and execute CLI-based or XML-based scripts or templates. The guide is intended for developers, skilled administrators, and operators who are expected to be familiar with the following: <ul style="list-style-type: none"> CLI scripting, XML, and the Velocity engine basic scripting or programming 5620 SAM functions
<i>5620 SAM Parameter Guide</i>	The <i>5620 SAM Parameter Guide</i> provides: <ul style="list-style-type: none"> parameter descriptions that include value ranges and default values parameter options and option descriptions parameter and option dependencies parameter mappings to the 5620 SAM-O XML equivalent property names There are dynamic links between the procedures in the <i>5620 SAM User Guide</i> and the parameter descriptions in the <i>5620 SAM Parameter Guide</i> . Parameters specific to LTE network elements are covered in the <i>5620 SAM LTE Parameter Reference</i> . Parameters specific to 1830 PSS network elements are covered in the <i>5620 SAM Optical Parameter Reference</i> .
<i>5620 SAM Statistics Management Guide</i>	The <i>5620 SAM Statistics Management Guide</i> provides information about how to configure performance and accounting statistics collection and how to view counters using the 5620 SAM. Network examples are included.

(2 of 4)

Guide	Description
<i>5620 SAM Maintenance Guide</i>	The <i>5620 SAM Maintenance Guide</i> provides procedures for: <ul style="list-style-type: none"> generating baseline information for 5620 SAM applications performing daily, weekly, monthly, and as-required maintenance activities for 5620 SAM-managed networks
<i>5620 SAM Troubleshooting Guide</i>	The <i>5620 SAM Troubleshooting Guide</i> provides task-based procedures and user documentation to: <ul style="list-style-type: none"> help resolve issues in the managed and management networks identify the root cause and plan corrective action for: <ul style="list-style-type: none"> alarm conditions on a network object or customer service problems on customer services with no associated alarms list problem scenarios, possible solutions, and tools to help check: <ul style="list-style-type: none"> network management LANs network management platforms and operating systems 5620 SAM client GUIs and client OSS applications 5620 SAM servers 5620 SAM databases
<i>5620 SAM Alarm Reference</i>	The <i>5620 SAM Alarm Reference</i> provides a list of all alarms that the 5620 SAM can raise. The reference is organized by network element type.
<i>5620 SAM Glossary</i>	The <i>5620 SAM Glossary</i> defines terms and acronyms used in all of the 5620 SAM documentation, including 5620 SAM LTE documentation.
<i>5620 SAM Network Element Compatibility Guide</i>	The <i>5620 SAM Network Element Compatibility Guide</i> provides release-specific information about the compatibility of managed device features in 5620 SAM releases.
5620 SAM LTE documentation	
<i>5620 SAM LTE RAN Release Description</i>	The <i>5620 SAM LTE RAN Release Description</i> provides information about the LTE RAN features associated with the release.
<i>5620 SAM LTE ePC User Guide</i>	The <i>5620 SAM LTE ePC User Guide</i> describes how to discover, configure, and manage LTE ePC devices using the 5620 SAM. The guide is intended for LTE ePC network planners, administrators, and operators. Alcatel-Lucent recommends that you review the entire <i>5620 SAM LTE ePC User Guide</i> before you attempt to use the 5620 SAM in your LTE network.
<i>5620 SAM LTE RAN User Guide</i>	The <i>5620 SAM LTE RAN User Guide</i> describes how to discover, configure, and manage the Evolved NodeB, or eNodeB, using the 5620 SAM. The guide is intended for LTE RAN network planners, administrators, and operators. Alcatel-Lucent recommends that you review the entire <i>5620 SAM LTE RAN User Guide</i> before you attempt to use the 5620 SAM in your LTE network.
<i>5620 SAM LTE Parameter Reference</i>	The <i>5620 SAM LTE Parameter Reference</i> provides a list of all LTE ePC and LTE RAN parameters supported in the 5620 SAM.
5620 SAM-O documentation	
<i>5620 SAM XML OSS Interface Developer Guide</i>	The <i>5620 SAM XML OSS Interface Developer Guide</i> provides information that allows you to: <ul style="list-style-type: none"> use the 5620 SAM XML OSS interface to access network management information learn about the information model associated with the managed network develop OSS applications using the packaged methods, classes, data types, and objects necessary to manage 5620 SAM functions
<i>5620 SAM 3GPP OSS Interface Developer Guide</i>	The <i>5620 SAM 3GPP OSS Interface Developer Guide</i> describes the components and architecture of the 3GPP OSS interface to the 5620 SAM. It includes procedures and samples to assist OSS application developers to use the 3GPP interface to manage LTE devices.

(3 of 4)

Guide	Description
<i>5620 SAM 3GPP OSS Interface Compliance Statements</i>	The <i>5620 SAM 3GPP OSS Interface Compliance Statements</i> document describes the compliance of the 5620 SAM 3GPP OSS interface with the 3GPP standard.
5620 SAM optical documentation	
<i>5620 SAM Optical User Guide</i>	The <i>5620 SAM Optical User Guide</i> describes how to discover, configure, and manage optical devices using the 5620 SAM. The guide is intended for optical network planners, administrators, and operators. Alcatel-Lucent recommends that you review the entire <i>5620 SAM Optical User Guide</i> before you attempt to use the 5620 SAM in your network.
<i>5620 SAM Optical Parameter Reference</i>	The <i>5620 SAM Optical Parameter Reference</i> provides a list of all optical device parameters supported in the 5620 SAM.

(4 of 4)

Obtaining customer documentation

You can obtain 5620 SAM customer documentation:

- from the product
- on the web

On-product documentation

The 5620 SAM on-product customer documentation is delivered in HTML and PDF. Choose Help→User Documentation from the 5620 SAM client GUI to open the help system in a web browser.

The help system opens to the User Documentation Index, which provides a summary of and links to all 5620 SAM customer documents.

Click on the Using the help system tab on the User Documentation Index page to find usage tips for navigating and searching within the on-product customer documentation.

You can return to the User Documentation Index at any time by clicking on the Home icon, shown in Figure 1.

Figure 1 Home icon



Documentation on the web

The 5620 SAM customer documentation is available for download in PDF format from the Alcatel-Lucent Customer Support Center: <http://www.alcatel-lucent.com/myaccess>. If you are a new user and require access to this service, please contact your Alcatel-Lucent support representative.

In addition to the guides listed in Table 1, Release Notices and other documents not delivered on-product are posted to this site.

Working with PDFs

You can download PDFs of individual guides from the Alcatel-Lucent Customer Support Center, or you can choose to download a zip of all PDFs for a particular release.

You can use the Search function of Acrobat Reader (File→Search) to find a term in a PDF of any 5620 SAM document. To refine your search, use appropriate search options (for example, search for whole words only or enable case-sensitive searching). You can also search for a term in multiple PDFs at once, provided that they are located in the same directory. For more information, see the Help for Acrobat Reader.

Cross-book hotlinks, for example, from a parameter name in the *5620 SAM User Guide* to a description of that parameter in the *5620 SAM Parameter Guide*, work only if both PDF files are in the same directory.



Note — Users of Mozilla browsers may receive an error message when opening the PDF files in the 5620 SAM documentation suite. The offline storage and default cache values used by the browsers are the cause of the error message.

Alcatel-Lucent recommends changing the Mozilla Firefox offline storage or Mozilla 1.7 cache value to 100 Mbytes to eliminate the error message.

Documentation conventions

Table 2 lists the conventions that are used throughout the documentation.

Table 2 Documentation conventions

Convention	Description	Example
Key name	Press a keyboard key	Delete
Italics	Identifies a variable	<i>hostname</i>
Key+Key	Type the appropriate consecutive keystroke sequence	CTRL+G
Key-Key	Type the appropriate simultaneous keystroke sequence	CTRL-G
*	An asterisk is a wildcard character, which means “any character” in a search argument.	log_file*.txt
↵	Press the Return key	↵
—	An em dash indicates there is no information.	—
→	Indicates that a cascading submenu results from selecting a menu item	Policies→Alarm Policies

Procedures with options or substeps

When there are options in a procedure, they are identified by letters. When there are substeps in a procedure, they are identified by Roman numerals.

Example of options in a procedure

At step 1, you can choose option a or b. At step 2, you must do what the step indicates.

- 1 This step offers two options. You must choose one of the following.
 - a This is one option.
 - b This is another option.
- 2 You must perform this step.

Example of substeps in a procedure

At step 1, you must perform a series of substeps within a step. At step 2, you must do what the step indicates.

- 1 This step has a series of substeps that you must perform to complete the step. You must perform the following substeps.
 - i This is the first substep.
 - ii This is the second substep.
 - iii This is the third substep.
- 2 You must perform this step.

Measurement conventions

Measurements in this document are expressed in metric units and follow the *Système international d'unités* (SI) standard for abbreviation of metric units. If imperial measurements are included, they appear in brackets following the metric unit.

Table 3 lists the measurement symbols used in this document.

Table 3 Bits and bytes conventions

Measurement	Symbol
bit	b
byte	byte
kilobits per second	kb/s

Important information

The following conventions are used to indicate important information:



Warning — Warning indicates that the described activity or situation may, or will, cause equipment damage or serious performance problems.



Caution — Caution indicates that the described activity or situation may, or will, cause service interruption.



Note — Notes provide information that is, or may be, of special interest.

Contents

Preface	ix
Prerequisites.....	ix
5620 SAM documentation suite	ix
Obtaining customer documentation	xii
On-product documentation.....	xii
Documentation on the web.....	xii
Documentation conventions.....	xiii
Procedures with options or substeps.....	xiii
Measurement conventions	xiv
Important information.....	xv

Getting started

1 —	Product and support overview	1-1
1.1	5620 SAM product overview	1-2
1.2	5620 SAM-O overview.....	1-2
	5620 SAM-O client.....	1-4
1.3	About this guide	1-4
1.4	Redundant network configurations.....	1-6
1.5	Supported devices and technologies	1-6
	Supported technologies	1-7
1.6	Before you begin	1-7
1.7	Open interfaces professional support.....	1-8

2 —	5620 SAM-O release features	2-1
2.1	5620 SAM-O Release 9.0 features and functions	2-2
2.2	Understanding changes, modifications, and deprecations to the 5620 SAM object model	2-4
2.3	Upgrade implications	2-5

5620 SAM-O communications

3 —	Communication with the 5620 SAM server	3-1
3.1	Overview	3-2
3.2	5620 SAM system components	3-2
3.3	Client interfaces	3-3
	Event monitoring using JMS	3-4
	Request and response using the 5620 SAM-O	3-4
	XML and SOAP	3-4
3.4	Secure communication	3-5
	SSL message encryption	3-5
	HTTPS communication with 5620 SAM-O	3-6
	Session management	3-6
	Procedure 3-1 To generate an MD5-hashed password for 5620 SAM main server access	3-7
	5620 SAM-O security features controlled through the 5620 SAM client GUI	3-8
3.5	5620 SAM-O set up and operation	3-9
	Procedure 3-2 To set up and operate the 5620 SAM-O	3-10
4 —	Event monitoring using JMS	4-1
4.1	Overview	4-2
4.2	JMS client planning considerations	4-2
4.3	JMS connections	4-2
4.4	Workflow to set up and operate a JMS connection	4-3
4.5	JMS subscriptions	4-4
	Client IDs	4-4
	JMS topics	4-4
	Filters	4-5
	Acknowledgement modes	4-5
	Persistence modes	4-6
4.6	JMS events	4-6
	JMS XML schemas	4-6
	JMS event header properties	4-7
	JMS event classes	4-9
4.7	JMS message filtering	4-23
	Mandatory events	4-24
	Simple message filters	4-24
	Optional properties	4-25
	Advanced message filters	4-26

4.8	JMS and redundancy	4-36
4.9	Connection monitoring and error recovery	4-37
	Procedure 4-1 To resync an OSS database with the 5620 SAM	4-38
	JMS exception	4-39
	Monitoring for incoming events	4-39
	5620 SAM-O session termination	4-40
	Missed events	4-40
4.10	Managing and monitoring JMS sessions	4-42
	Monitoring sessions	4-42
	JMS logging.....	4-43
	Procedure 4-2 To manage a 5620 SAM-O JMS client session or remove a durable subscription.....	4-43
4.11	JMS consumer creation.....	4-43
	Procedure 4-3 To compile and create a sample JMS client *.java file	4-44
4.12	JMS record and playback tool	4-46
	Procedure 4-4 To record, stop and playback JMS messages.....	4-46
5 —	XML requests	5-1
5.1	XML communication with the 5620 SAM server	5-2
	PostXML	5-2
	Procedure 5-1 To post an XML request to the 5620 SAM server	5-4
5.2	Monitoring the status of the 5620 SAM-O connection using XML API ping	5-5
	XML API ping	5-5
5.3	Workflow to set up and operate an HTTP XML request-response connection	5-6
5.4	Viewing XML requests ignored by the server.....	5-7
	Procedure 5-2 To enable server log file logging of ignored XML requests.....	5-7
5.5	5620 SAM-O server-specific commands	5-8
	Obtaining the local server time for a request	5-8
	Identifying the server software load	5-8
 5620 SAM-O information model		
6 —	5620 SAM-O information model overview	6-1
6.1	5620 SAM-O information model overview	6-2
6.2	5620 SAM-O XML packages	6-3
7 —	5620 SAM-O XML Reference	7-1
7.1	Overview	7-2
7.2	Main menu.....	7-3
7.3	Package list	7-4
7.4	Classes and types list	7-4
7.5	Class details.....	7-5
	Description	7-5
	Inheritance hierarchy.....	7-5

	Direct subclass.....	7-5
	Parent hierarchy	7-6
	Children hierarchy	7-6
	Relationships.....	7-6
	Class type.....	7-7
	Naming.....	7-7
	Methods returning children.....	7-7
	Stats.....	7-8
	Properties	7-9
	Methods.....	7-10
	Supported network elements.....	7-12
7.6	Type details	7-13
7.7	XML schema changes	7-13
	Deprecated properties and methods	7-13
7.8	JMS changes.....	7-14
7.9	Procedures	7-14
	Procedure 7-1 To display the 5620 SAM-O XML Reference	7-14
	Procedure 7-2 To view release-specific XML schema changes	7-14
	Procedure 7-3 To view release-specific JMS changes.....	7-15
	Procedure 7-4 To view deprecated schema items	7-15
	Procedure 7-5 To view the general methods and types	7-16
	Procedure 7-6 To view the supported device types.....	7-16
8 —	5620 SAM-O XML Schema	8-1
8.1	5620 SAM-O XML Schema overview	8-2
	Procedure 8-1 To display specific 5620 SAM-O XML package information.....	8-2
	XML standard and specific schema files	8-2
	XML package methods schema files	8-7
	XML package types schema files	8-8
	Common elements of classes (types)	8-9
9 —	XML message structure	9-1
9.1	XML message structure overview	9-2
	Message structure	9-2
	Requests	9-4
	Responses.....	9-10
	Faults and exceptions	9-11
	Log file for XML requests, responses, and exceptions.....	9-15
9.2	CLI commands within XML methods	9-17
9.3	Mapping XML methods to GUI operations.....	9-18
	Procedure 9-1 To enable logging of GUI operations on the 5620 SAM server	9-18
	Procedure 9-2 To view the GUI operation in the server log	9-19
	Output format of a GUI operation in the server log file	9-19

OSS domains

10 –	Fault management	10-1
10.1	Fault management overview	10-2
10.2	Communicating with the 5620 SAM.....	10-2
10.3	Alarm sources.....	10-2
	Network traps.....	10-2
	5620 SAM alarms.....	10-3
10.4	Alarm definitions.....	10-3
10.5	Alarm messages	10-3
10.6	Retrieving alarms	10-6
10.7	Alarm management	10-6
10.8	Alarm policies.....	10-7
10.9	Alarm correlation	10-7
10.10	OSS client alarm testing.....	10-7
	Test alarms.....	10-7
	JMS record and playback.....	10-8
10.11	Workflow to set up alarm management	10-9
11 –	OAM	11-1
11.1	OAM overview.....	11-2
11.2	Key packages and classes	11-2
11.3	Tests	11-2
11.4	Test suites.....	11-3
11.5	Generated tests	11-4
11.6	Workflow to generate tests.....	11-4
11.7	OAM Test Results file retrieval.....	11-4
	OAM log retrieval example using the <registerSasLogToFile> method.....	11-6
12 –	Inventory management	12-1
12.1	Inventory management overview	12-2
12.2	Inventory retrieval	12-2
	Concurrent find or findToFile requests	12-4
	find method.....	12-4
	findToFile method	12-4
	Procedure 12-1 To configure remote findToFile result storage	12-7
12.3	Filtering.....	12-9
	Children filters	12-10
	Filter element types	12-11
12.4	Network discovery	12-15
	5620 SAM object hierarchy	12-15
12.5	Guidelines for inventory discovery and maintenance	12-16
13 –	Accounting and performance monitoring	13-1
13.1	Accounting and performance monitoring overview	13-2
13.2	References	13-2

13.3	Key packages and classes	13-2
	log package	13-2
	accounting package	13-3
	file package	13-3
	root package	13-3
13.4	Statistic objects	13-3
13.5	Workflow to retrieve statistics data	13-4
13.6	Scheduled and on-demand statistics	13-4
13.7	Statistics retrieval methods	13-6
	Procedure 13-1 To configure the registerLogtoFile storage requirements.....	13-6
13.8	Statistics retrieval using registerLogToFile	13-6
	Statistics recovery after OSS connection loss.....	13-10
	Statistics recovery after JMS client unsubscribed.....	13-10
13.9	Statistics retrieval using findToFile	13-10
	Statistics retrieval examples using the findToFile method	13-11
	Statistics recovery after OSS connection loss.....	13-12
	Missing performance statistics	13-12
	XML statistics output file	13-12
13.10	Statistics monitoring using JMS	13-14
13.11	Collecting JMS performance statistics	13-14
13.12	Sample request to retrieve historical statistics from an NE.....	13-15
13.13	Sample request to collect interface statistics.....	13-15

Configuration management

14 —	Configuration management overview	14-1
14.1	Configuration management overview	14-2
14.2	Configuration methods	14-2
	Modify example: To configure an existing port.....	14-3
	Create example: To create an Epipe object	14-5
	Delete example: To delete an Epipe object	14-6
14.3	Deployers.....	14-7
	Deployment types <deployer>.....	14-7
	Synchronous and asynchronous requests	14-8
	Deployer failures	14-10
	Deployer failure recovery procedures	14-12
	Deployer alarms and events	14-12
	Deployer simulation.....	14-14
14.4	Configuration management guidelines for multiple requests	14-15
14.5	Workflow to handle deployer failures	14-16
15 —	Device configuration management	15-1
15.1	Device configuration overview	15-2
15.2	Workflow to configure equipment.....	15-2
15.3	Generic NE profiles	15-3

15.4	Generic NE profile parameter configuration.....	15-3
15.5	Device software upgrades	15-4
16	Network configuration management	16-1
16.1	Network configuration management overview	16-2
16.2	Network interface configuration.....	16-2
16.3	Workflow to configure network interfaces	16-2
16.4	Static route configuration	16-3
16.5	Routing protocol configuration	16-3
	BGP	16-4
	RIP	16-4
	OSPF	16-4
	LDP	16-4
	IS-IS.....	16-5
	L2TP	16-5
	RSVP	16-5
	PIM.....	16-5
	IGMP	16-5
	MSDP	16-5
	MLD.....	16-6
16.6	Workflow to configure a routing protocol.....	16-6
16.7	VRRP virtual router configuration	16-7
16.8	Workflow to configure a virtual router	16-8
16.9	MPLS, LSP, and service tunnel configuration	16-8
	MPLS.....	16-8
	LSP.....	16-8
	Service tunnels	16-9
16.10	Workflow to configure an MPLS path, LSP, and service tunnel	16-9
17	Policy configuration management	17-1
17.1	Policy configuration overview.....	17-2
	Service management policies	17-2
	Routing management policies	17-4
	Network management policies	17-4
	Policy distribution mode	17-5
	Policy configuration mode	17-6
17.2	Workflow to configure ACL IP and MAC filter policies.....	17-7
17.3	Workflow to configure routing management policies	17-7
18	Service configuration management	18-1
18.1	Service configuration overview	18-2
	VLL	18-2
	VPLS/HVPLS/MVPLS.....	18-4
	IES	18-7
	VPRN	18-9
	VLAN.....	18-12
	Mirror	18-13
	Optical transport service.....	18-15
	Object life cycle.....	18-16

18.2	Workflow to configure a mirror service	18-16
18.3	Workflow to configure an optical transport service	18-17
18.4	Composite services	18-17
	Network discovery of composite services	18-19
	Connector types	18-19
	Sample composite service configuration	18-21
18.5	Workflow to configure a composite service.....	18-22
18.6	IGMP snooping	18-22
18.7	Workflow to configure IGMP snooping	18-22
18.8	DHCP relay configuration	18-23
18.9	Workflow to configure DHCP relay	18-23
18.10	IPsec management	18-23
	Typical IPsec configuration example	18-24
	5620 SAM-O IPsec configuration.....	18-24
	OSSI configuration requirements for static tunnel type.....	18-24
	OSSI requirements for dynamic tunnel type	18-24
18.11	Workflow to configure IPsec – option A.....	18-25
18.12	Workflow to configure IPsec – option B.....	18-25
18.13	Customer and residential subscriber configuration overview	18-26
	Customer configuration	18-26
	Residential subscriber	18-26
18.14	Workflow to configure and manage residential subscribers.....	18-27

19 – Script and template configuration management 19-1

19.1	Overview	19-2
19.2	Script management	19-2
19.3	Workflow to execute a script	19-3
19.4	XML API template configuration management.....	19-3

Integrated products

20 – 5650 CPAM integration configuration 20-1

20.1	5650 CPAM integration overview	20-2
20.2	5650 CPAM OSS interface overview.....	20-4
	Multiple topologies.....	20-4
	5650 CPAM administrative domains	20-5
	Topology checkpoints.....	20-5
	Route management	20-8
	Path monitoring	20-10
	Fault management	20-10
20.3	5650 CPAM OSS packages	20-11

Appendices

A.	Troubleshooting	A-1
A.1	Troubleshooting client OSS application problems.....	A-2
	Procedure A-1 The client OSS application cannot communicate with the server.....	A-2
	Procedure A-2 An attempt to log in to the 5620 SAM server fails	A-3
	Procedure A-3 Receive insufficient privileges to perform this operation on an object exception when performing an action on a 5620 SAM object	A-4
	Procedure A-4 An XML request fails	A-4
	Procedure A-5 Unable to perform an action using the 5620 SAM-O	A-4
	Procedure A-6 Identifying XML messages from specific users	A-5
	Procedure A-7 Receive a java.lang.UnsupportedClassVersionError when sending scripts using the PostXML tool	A-6
	Procedure A-8 Receive a java.net.ConnectException when sending scripts using the PostXML tool.....	A-7
	Procedure A-9 Receive a java.net.ConnectException when sending HTTP requests to the 5620 SAM-O server	A-8
	Procedure A-10 The OSS client cannot connect with the HTTP or JMS server	A-8
	Procedure A-11 The OSS client cannot perform find or findToFile requests.....	A-9
B.	5620 SAM-O SDK library of samples	B-1
B.1	Description 5620 SAM-O SDK library of XML samples	B-2
	Basic SDK	B-2
	Advanced SDK	B-2

Getting started

- 1 — Product and support overview
- 2 — 5620 SAM-O release features

1 — *Product and support overview*

- 1.1 5620 SAM product overview 1-2**
- 1.2 5620 SAM-O overview 1-2**
- 1.3 About this guide 1-4**
- 1.4 Redundant network configurations 1-6**
- 1.5 Supported devices and technologies 1-6**
- 1.6 Before you begin 1-7**
- 1.7 Open interfaces professional support 1-8**

1.1 5620 SAM product overview

The 5620 SAM is a network management system that is designed to manage Alcatel-Lucent network elements, or NEs, such as routers and switches. The 5620 SAM also supports the management of some Telco devices, and provides limited management of other third-party devices called generic NEs.

The 5620 SAM network management key features include:

- alarm correlation up to the service level
- service and routing provisioning using policies and profiles to reduce the repetition of effort
- inventory reporting at the equipment, service, and customer levels using filtered views that can be saved as report files
- network performance and accounting data collection on a per-service or per-port basis using a flat-rate, destination, or usage schema
- troubleshooting at the service level, which includes viewing the associations between services and entities such as customers, equipment, and transport tunnels
- OSS interfaces that provide access to the 5620 SAM functions from other systems

The 5620 SAM software architecture is built on industry standards including Java and J2EE framework, multi-tier layering, and web service, XML/SOAP and 3GPP interfaces. The use of industry-standard interfaces allows the 5620 SAM to interoperate with other network management systems.

The main components of the 5620 SAM system includes a Java-based SAM server, an Oracle database server that provides consistent storage of network data, Java-based 5620 SAM GUI clients, and platform-independent OSS clients.

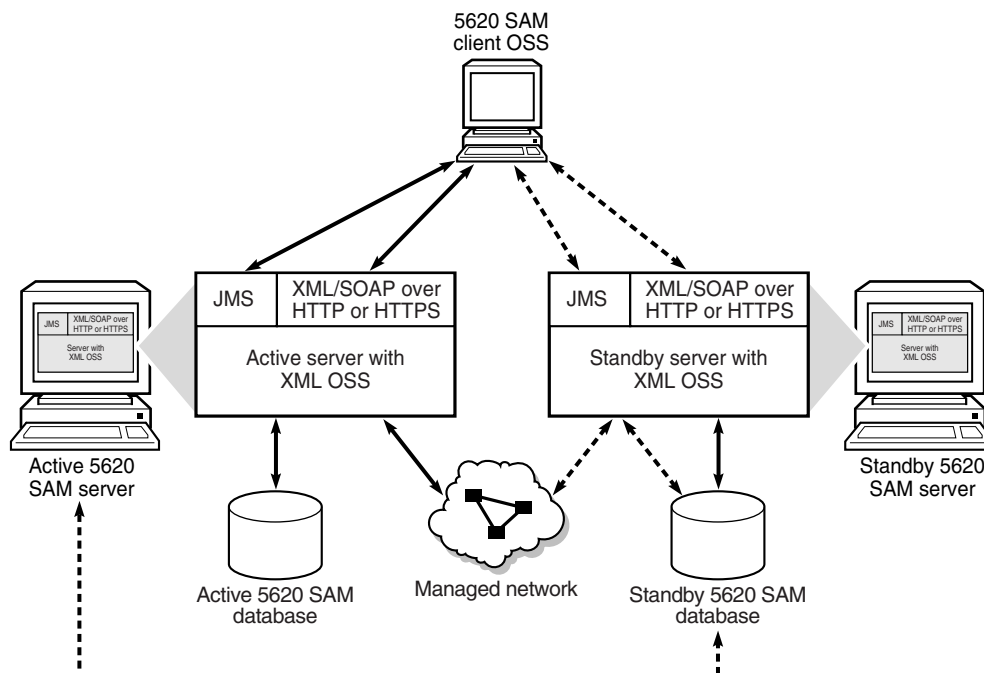
1.2 5620 SAM-O overview

The 5620 SAM-O is a 5620 SAM module that provides an XML interface to the 5620 SAM through which an OSS client application can perform the following tasks:

- configure or retrieve 5620 SAM network management information
- manipulate managed objects
- receive event notifications from the 5620 SAM server using a persistent or non-persistent JMS connection

Figure 1-1 shows the architecture of the 5620 SAM-O, which includes a JMS connection for the collection of near-real-time alarms and events.

Figure 1-1 5620 SAM-O architecture



17702

A 5620 SAM-O client sends an XML-encoded, SOAP-enabled message to a port on the HTTP or HTTPS server that runs on a 5620 SAM main server. The default HTTP port for incoming messages is 8080. An XML application at port 8080 parses the SOAP XML message and sends the request to the 5620 SAM.



Caution 1 – The 5620 SAM-O has a limit of 10 concurrent and active HTTP connections.

Caution 2 – A limit of 10 concurrent XML/SOAP requests can be run over the 10 HTTP connections.

Alcatel-Lucent strongly recommends that you use only one HTTP connection for your XML/SOAP requests.

If more than one HTTP connection is required, you must make the number of HTTP connections configurable at run time so that system integrators can manage the HTTP connections across the OSS applications that connect to the same 5620 SAM server. Contact Alcatel-Lucent OIPS technical support for more information.

The 5620 SAM-O interface is installed with the core 5620 SAM software components. To use the 5620 SAM-O interface, you must perform the following configuration tasks.

- Enter a 5620 SAM license key that activates the 5620 SAM-O.
- Use a user group that is assigned the OSS management scope of command role.



Note — You can create a scope of command profile to group the OSS management scope of command role with additional roles. See the *5620 SAM User Guide* for more information about 5620 SAM user groups and the associated scope of command roles and permissions.

5620 SAM-O client

The 5620 SAM-O client implementation can range from a CLI to a third-party application. The format for the client user interface varies with the function of the client application; for example, rolling up statistics into a third-party billing application. The 5620 SAM-O clients function the same way regardless of the front-end implementation: XML scripts are posted to the 5620 SAM server using an HTTP client. See section 3.1 for more information.

1.3 About this guide

The *5620 SAM XML OSS Interface Developer Guide* provides information to help OSS developers create applications that perform 5620 SAM functions. It is part of the 5620 SAM-O documentation set, which also includes the *5620 SAM-O XML Reference* and *5620 SAM-O Schema*.

This guide contains the following volumes:

- Getting started—contains an overview of the 5620 SAM-O interface and includes the following:
 - a 5620 SAM and 5620 SAM-O high-level overview
 - redundant configurations and supported devices and technologies
 - open interfaces professional support
- 5620 SAM-O communications—contains information about how an OSS via the 5620 SAM-O interface, can communicate with the 5620 SAM and the underlying network, including the following:
 - communication with the 5620 SAM server
 - event monitoring
 - XML requests
 - interworking with other systems
- 5620 SAM-O information model—contains information about the 5620 SAM-O object model that includes the following:
 - XML package descriptions, object properties, and value definitions and relationships
 - XML Reference and XML Schema type and method definitions and descriptions
 - XML message structure and request/response operation

- OSS domains—contains information that a developer can use to help with the development of applications for the following OSS domains:
 - Fault management—provides information for developing an event and alarm management system
 - Inventory management—provides information about inventory retrieval
 - Accounting and Performance Management
 - OAM diagnostic tests
- Configuration management—contains information about how an OSS can be used to make changes to network objects, which can involve creating, deleting, and modifying the objects in the following areas:
 - general configuration using generic method that applies to all areas
 - equipment configuration
 - routing protocol configuration
 - customer and residential subscriber configuration
 - policy configuration
 - service configuration
 - XML service template configuration
 - script management configuration
- Integrated products—contains information about other 5620 SAM network management products, such as:
 - 5650 CPAM
- Appendices—contains information about the following:
 - troubleshooting client OSS application problems
 - 5620 SAM-O SDK library samples

The following guides, documents, or files are also of interest to an OSS developer:

- When setting up the 5620 SAM in your lab environment:
 - *5620 SAM Planning Guide*—contains information about supported platforms
 - *5620 SAM | 5650 CPAM Installation and Upgrade Guide* —includes installation instructions
- When starting to use the 5620 SAM:
 - *5620 SAM User Guide*—contains GUI basics and network configuration information
- When developing your OSS:
 - *5620 SAM-O XML Reference*—contains javadoc-style documentation of the object model, essential for writing XML API requests
 - *5620 SAM-O Schema*—contains XSD schema definitions for the object model
 - JMS Example Code—contains example code for writing JMS subscribers
- Other files:
 - MIBs—files can be found on the 5620 SAM server under <install directory>/nms/web/tomcat/work/Catalina/localhost/help/eclipse/plugins/ALU_SAM_MIB or on the DVD under nms/installer/install_point/User_Documentation/5620_SAM-O_documentation/XML_Reference/mib.

1.4 Redundant network configurations

The 5620 SAM server manages all transactions in the 5620 SAM database. The 5620 SAM supports database and server redundancy, which provides network visibility if hardware or software fails on one or more 5620 SAM components; for example, a server. If the active component fails, the standby component is configured to automatically take control. You can also perform a manual switch to test redundancy or to prepare for a network upgrade.

The 5620 SAM product suite supports collocated and distributed redundant configurations. In a collocated installation, the 5620 SAM server and database are on the same workstation. In a distributed installation, the 5620 SAM server and the 5620 SAM database are on separate workstations. The servers and databases are four separate entities regardless of the physical or geographical deployment.



Note — Redundant network configuration is only supported on Solaris workstations.

You must know the following information to configure redundant 5620 SAM connections:

- the IP addresses of the primary and standby main servers
- which main server is currently primary

You can use any of the following methods to monitor the connection status between the 5620 SAM OSS client and the active server:

- JMS exception to monitor client connections to the JMS server
- monitoring near-real-time events using JMS client connections to the JMS server
- XML API ping to monitor the HTTP connection to the 5620 SAM web server

See section [3.3](#) for more information about monitoring the connection status between the 5620 SAM OSS client and the active server.

See chapter [4](#) for more information about JMS and redundancy.

1.5 Supported devices and technologies

For core-network devices such as the 7450 ESS, 7710 SR, and 7750 SR, the 5620 SAM supports the current software release and the three immediately previous major releases. For example, a Release 9.0 5620 SAM system can manage a 7750 SR at Release 9.0, 8.0, 7.0, 6.1, or 6.0. Older device releases are not supported.

For devices other than core-network devices, the 5620 SAM release support varies. See the *5620 SAM Network Element Compatibility Guide* for specific device support information.



Caution — When a device introduces a maintenance release, you must ensure that your 5620 SAM software supports the new device release before you upgrade the device to the new release. You may need to upgrade the 5620 SAM to support the new device release. See the current *5620 SAM Network Element Compatibility Guide* for more information.



Note — The 5620 SAM functions and features vary for the supported devices. The *5620 SAM XML OSS Interface Developer Guide* does not describe the variations in support, or device-specific procedures for specific implementations. See the *5620 SAM User Guide* for this information, and the *5620 SAM Parameter Guide* for device-specific parameters, default parameter values, and parameter ranges.

The following documents describe 5620 SAM support for specific device releases:

- *5620 SAM Network Element Compatibility Guide*—lists the device releases and functions that the 5620 SAM supports
- *5620 SAM Parameter Guide*—lists and describes the supported parameters; includes device-specific variations in parameter defaults and ranges

See the device user documentation for information about device functions, parameters, and CLI commands that are outside the scope of the 5620 SAM documentation. Contact Alcatel-Lucent technical support for specific network or facility considerations.

Supported technologies

The 5620 SAM-O uses the following technologies:

- | | |
|-----------|---------|
| • XML | • Java |
| • SOAP | • JMS |
| • HTTP(S) | • JBoss |

See the *5620 SAM System Architecture Guide* for more information about the standards compliance and versions of these technologies.

1.6 Before you begin

Alcatel-Lucent recommends that you do the following before you start to work in the 5620 SAM-O XML development environment.

- Review the requirements for license key configuration and platform size considerations. See the *5620 SAM Planning Guide* for more information.
- Preconfigure the device that you want the 5620 SAM to manage. See the device documentation and the *5620 SAM User Guide* for more information.

- Back up the 5620 SAM database. Alcatel-Lucent recommends a database backup before you implement any major, network-wide configuration changes. A database backup utility is available in the 5620 SAM client GUI and through a CLI. See the *5620 SAM User Guide* for more information.
- Understand OSS interfaces and standard XML schema constructions

Contact your Alcatel-Lucent OIPS representative for development support. See section 1.7 for more information.

1.7 Open interfaces professional support

Alcatel-Lucent provides developer support for third-party application integration with the 5620 SAM-O. The Alcatel-Lucent OIPS portfolio provides OSS developers with network management integration solutions for the 5620 SAM and 5650 CPAM. OSS integration initiatives include project review, design consultation, development support, and training for integration projects.

Key elements of each service offering include:

- Design consultation and high-level project reviews. Extensive software design consultation provides architectural review of a proposed design and guidance on whether the design methodology is suited to the proposed application given Alcatel-Lucent experience with the platform and interface.
- Software development support and technical guidance
- Sample code that provide examples of configuration and management concepts, to facilitate rapid OSS integration
- OSS product training that minimizes project down time and optimizes development efficiency
- OSS Product Certification support to ISVs identified as Connected Partners, involving extensive interoperability testing of OSS applications with the 5620 SAM and 5650 CPAM.

Alcatel-Lucent OIPS can customize the support content and contract length to meet the unique development requirements of service providers.

2 — 5620 SAM-O release features

- 2.1 5620 SAM-O Release 9.0 features and functions 2-2**
- 2.2 Understanding changes, modifications, and deprecations to the 5620 SAM object model 2-4**
- 2.3 Upgrade implications 2-5**

2.1 5620 SAM-O Release 9.0 features and functions

Table 2-1 lists the features and functions added in the 5620 SAM Release 9.0 for 5620 SAM-O support. See the other chapters and appendixes in this guide for more information about 5620 SAM-O features and functions. See the *5620 SAM User Guide* for information about non-5620 SAM-O features and functions.

Table 2-1 5620 SAM-O Release 9.0 features, problem fixes, and documentation changes

Feature or function	Description	Reference for more information
Release 9.0 R7 features		
No features have been added in this release.		
Release 9.0 R7 problem fixes		
Enumerated forwarding class values for OAM tests	In test results sent to the OSSl, the 5620 SAM returns the numeric forwarding class value in addition to the alphanumeric identifier.	See the <i>5620 SAM-O XML Reference</i>
Release 9.0 R7 documentation content enhancements		
Configuration management	Miscellaneous updates to the chapters.	See chapters 14 , 15 , 16 , 17 , 18 , and 19 .
Release 9.0 R7 documentation structure changes		
Template management	Consolidated template management information into the script management chapter.	See chapter 19 .
Residential subscriber configuration	Consolidated residential subscriber configuration information into the service management chapter.	See chapter 18 .
Release 9.0 R6 features		
No features have been added in this release.		
Release 9.0 R6 problem fixes		
No problem fixes have been added in this release.		
Release 9.0 R6 documentation content enhancements		
No documentation content enhancements have been added in this release.		
Release 9.0 R6 documentation structure changes		
No documentation structure changes have been added in this release.		
Release 9.0 R5 features		
Messaging rates for durable JMS clients	Because of changes in the mechanism used to queue JMS messages, the messaging rates for durable JMS clients have changed.	See the <i>5620 SAM Planning Guide</i> .
Advanced JMS message filters	You can create advanced filters to reduce the number of JMS event messages sent to OSS clients. An advanced JMS message filter is a 5620 SAM-O XML filter that specifies the following: <ul style="list-style-type: none"> the types of event notifications that are to be sent to the client a set of object properties to use as match criteria for messages sent to the client 	See chapter 4 and the <i>5620 SAM Statistics Management Guide</i> .

(1 of 3)

Feature or function	Description	Reference for more information
registerLogToFile supports performance statistics export	You can use the 5620 SAM-O registerLogToFile method to export performance statistics, in addition to accounting statistics.	See chapter 13.
registerSasLogToFile supports retrieval of OAM logs	You can use the 5620 SAM-O registerSasLogToFile method to retrieve OAM logs that are based on specific class types.	See chapter 11.
Release 9.0 R5 problem fixes		
No problem fixes have been added in this release.		
Release 9.0 R5 documentation content enhancements		
Fault management	Miscellaneous updates to the chapter.	See chapter 10.
OAM	Miscellaneous updates to the chapter.	See chapter 11.
Inventory management	Miscellaneous updates to the chapter.	See chapter 12.
XML Reference	Miscellaneous updates to the chapter.	See chapter 7.
Release 9.0 R5 documentation structure changes		
No documentation structure changes have been added in this release.		
Release 9.0 R3 documentation content enhancements		
Configuration management	Miscellaneous updates to the section.	See section 14.1.
5620 SAM-O information model	Miscellaneous updates to the chapter.	See chapter 6.
Product and support	Miscellaneous updates to the chapter.	See chapter 1.
Release 9.0 R3 documentation structure changes		
—	Moved the OAM information from the Fault Management chapter to a standalone chapter in the OSS Domains volume.	See chapter 11.
—	Removed Appendix B and placed the information in another chapter.	See chapter 13.
Release 9.0 R1 features		
Optical transport service support	The 5620 SAM supports optical transport service configuration. The API is new and evolving, and is subject to change in future releases.	See “Optical transport service” in chapter 18.
Release 9.0 R1 problem fixes		
No problem fixes have been added in this release.		
Release 9.0 R1 documentation content enhancements		
Generic methods	Added information about the following: Generic methods for error recovery include the generic.GenericObject.getDeployersShallow method.	See section 14.3.
Filter construction	Added information about the following: Filter construction requires that you know the class to which you want to apply the filter. The calls can be defined by the class attribute in the top-level <filter> element.	See section 12.3.
Methods for configuring objects	Added references to generic methods and object hierarchy.	—

(2 of 3)

Feature or function	Description	Reference for more information
Output format of a GUI operation in the server log file	Updated the subsection.	See section 9.3.
Exceeding find or findToFile request limit	Added information about the following: If 5 concurrent <find> or <findToFile> requests is reached, subsequent <find> or <findToFile> requests are rejected with an XMLException or IMException response respectively.	See section 12.2.
Release 9.0 R1 documentation structure changes		
No documentation structure changes have been added in this release.		

(3 of 3)

2.2 Understanding changes, modifications, and deprecations to the 5620 SAM object model

Each major and minor 5620 SAM-O release includes a schema changes file that documents the following activities, release-over-release and between releases:

- objects, such as classes, removed or changed from the object model
- packages, such as netw, generic, and snmp, that are added or removed from the object model
- types, such as bitmasks and enumerations, that are added, changed, or removed from the object model
- parameters, such as logFullAction, that are removed or added to classes in the object model

You can use the information in the Schema Changes file to:

- ensure that existing third-party or in-house OSS applications correctly reference the latest 5620 SAM-O object model
- track changes to the object model as third-party or in-house OSS applications are developed using different 5620 SAM-O releases



Note — See the *5620 SAM-O XML Reference Schema Changes* file and *5620 SAM XML OSS Interface Developer Guide* from the current and previous releases to assess the effect of interface updates.

The following is a condensed sample of the Schema Changes file contents from a previous release:

Release 5.0 R4 to 5.0 R5

Packages Added

The following packages are new: schedule

Objects Removed

Types:


```

    sas.Scope of type: Enumeration

Objects Added

Types:

    sas.TestGroupRole of type: Enumeration

    sas.ExecutionStrategy of type: Enumeration

    Classes:

    mpls.LspTraceDefinition

    mpls.LspPingDefinition

Objects Changed

Types:

    sas.ReturnCode of type: Enumeration

    Added enum downstreamNotInMfib

    Classes:

    mpls.Interface

    Added relativeName [interface-$interfaceId]

    Added property mpls.ActualHop-Set

```

2.3 Upgrade implications

When you upgrade the 5620 SAM, the FDNs may change. 5620 SAM-O clients that use FDNs to reference 5620 SAM objects must update their FDNs accordingly. The FDNs that change during an upgrade are logged to files in the 5620 SAM database installation directory. One log file is created for each upgrade transition in which an FDN changes in the full upgrade path.

For example, if you upgrade from the 5620 SAM, Release 8.0 R1, to the 5620 SAM, Release 9.0 R1, the upgrade process creates FDN mapping log files that have the following names:

- FDN_Mapping_8_0_R1_to_8_0_R2.*date*.log
- FDN_Mapping_8_0_R2_to_8_0_R3.*date*.log
- FDN_Mapping_8_0_R3_to_8_0_R4.*date*.log
- FDN_Mapping_8_0_R4_to_8_0_R5.*date*.log
- FDN_Mapping_8_0_R5_to_8_0_R6.*date*.log
- FDN_Mapping_8_0_R6_to_8_0_R7.*date*.log
- FDN_Mapping_8_0_R7_to_8_0_R8.*date*.log
- FDN_Mapping_8_0_R8_to_9_0_R1.*date*.log

where *date* is the date of the upgrade



Note — A 5620 SAM system upgrade automatically migrates data from the start to the end release without the need to upgrade to each intermediate release.

Code 2-1 shows an example of the contents of an FDN mapping log file.

Code 2-1: Sample FDN mapping log file

```
<?xml version='1.0' ?>
<version from="A.0.R1" to="B.0.R1">
  <old
Fdn="network:142.165.76.50:router-1:ospf:areaSite-0.0.0.11:interface
-204.83.242.2.0" class="ospf.Interface">
    <new
Fdn="network:142.165.76.50:router-1:ospf-v2:areaSite-0.0.0.11:interf
ace-204.83.242.2.0" class="ospf.Interface"/>
    </old>
  </old>
  <old
Fdn="network:142.165.76.50:router-1:ospf:areaSite-0.0.0.11:interface
-204.83.242.2.0:neighbor-204.83.242.1.0" class="ospf.Neighbor">
    <new
Fdn="network:142.165.76.50:router-1:ospf-v2:areaSite-0.0.0.11:interf
ace-204.83.242.2.0:neighbor-204.83.242.1.0" class="ospf.Neighbor"/>
    </old>
  </old>
  <old
Fdn="network:142.165.76.50:router-1:ospf:areaSite-0.0.0.11:interface
-204.83.242.2.0:md 5-1" class="ospf.Md5Key">
    <new
Fdn="network:142.165.76.50:router-1:ospf-v2:areaSite-0.0.0.11:interf
ace-204.83.242.2.0:md5-1" class="ospf.Md5Key"/>
    </old>
  </old>
  <old
Fdn="network:142.165.76.50:router-1:ospf:areaSite-0.0.0.11:interface
-204.83.242.9.0" class="ospf.Interface">
    <new
Fdn="network:142.165.76.50:router-1:ospf-v2:areaSite-0.0.0.11:interf
ace-204.83.242.9.0" class="ospf.Interface"/>
    </old>
  </old>
  <old
Fdn="network:142.165.76.50:router-1:ospf:areaSite-0.0.0.11:interface
-204.83.242.9.0:md5-1" class="ospf.Md5Key">
    <new
Fdn="network:142.165.76.50:router-1:ospf-v2:areaSite-0.0.0.11:interf
ace-204.83.242.9.0:md5-1" class="ospf.Md5Key"/>
    </old>
  </old>
</version>
```

5620 SAM-O communications

- 3 – Communication with the 5620 SAM server
- 4 – Event monitoring using JMS
- 5 – XML requests

3 — *Communication with the 5620 SAM server*

- 3.1 Overview 3-2**
- 3.2 5620 SAM system components 3-2**
- 3.3 Client interfaces 3-3**
- 3.4 Secure communication 3-5**
- 3.5 5620 SAM-O set up and operation 3-9**

3.1 Overview

An OSS can communicate with the 5620 SAM through the following 5620 SAM interfaces:

- the JMS event stream and SOAP XML API, which are collectively called the 5620 SAM XML OSS interface
- the 3GPP OSS interface

The 5620 SAM XML OSS interface receives the JMS event stream, which is used for near-real-time event notification of changes in the network. The XML API is used to change configurations and to request status information.

The 3GPP OSS interface is a CORBA-based API that provides a 3GPP-compliant layer for client application communication with a 5620 SAM-managed LTE network. See the *5620 SAM 3GPP OSS Interface Developer Guide* for more information about the 5620 SAM 3GPP OSS interface.

Developers that create OSS applications are expected to be familiar with the following:

- Standard XML schema constructions
- OSS interfaces
- Managed devices
- 5620 SAM functionality.

The following sections describe the main components of the 5620 SAM system and the interfaces in the 5620 SAM server communications with the 5620 SAM-O clients.

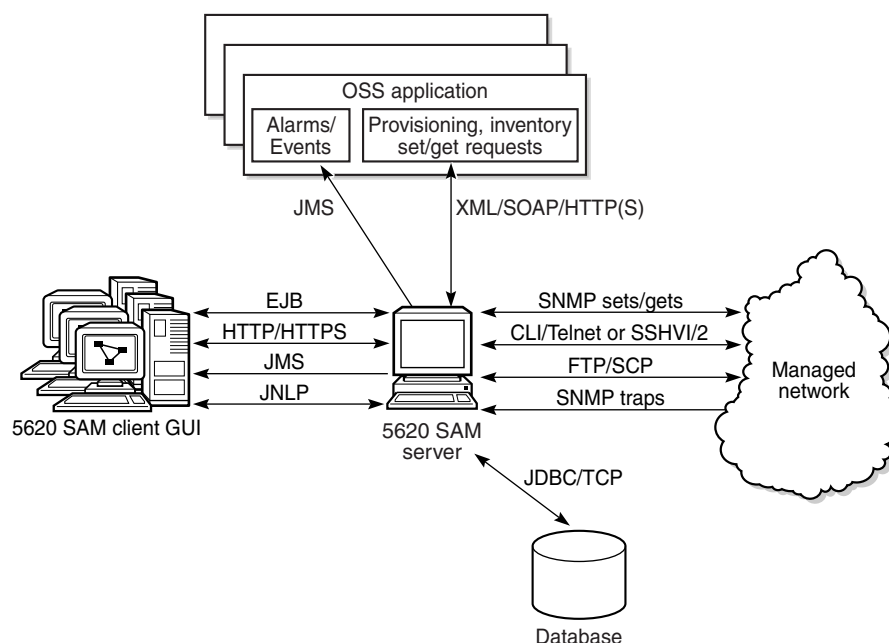
3.2 5620 SAM system components

The following are the main components of a 5620 SAM system.

- The main server is a network-management processing engine written in Java that runs on a Solaris platform. The server includes several third-party components, such as an application server, JMS server, web server, protocol stack set, and database adaptor. Server functionality can be concentrated on one physical platform, or distributed across multiple dedicated stations.
- The clients are OSS applications or Java-based 5620 SAM GUIs. The OSS applications can run on any platform, because they use platform-neutral protocols to communicate with the 5620 SAM main server. The 5620 SAM GUI and 5620 SAM-O clients do not communicate directly with auxiliary servers.
- The database is a customized Oracle relational database that provides persistent storage for the network data. The database can run on the same station as the 5620 SAM main server software, or on a different station.

The 5620 SAM component interfaces use industry-standard protocols for communication between servers and the database, managed network devices, and clients, as shown in Figure 3-1.

Figure 3-1 5620 SAM interfaces



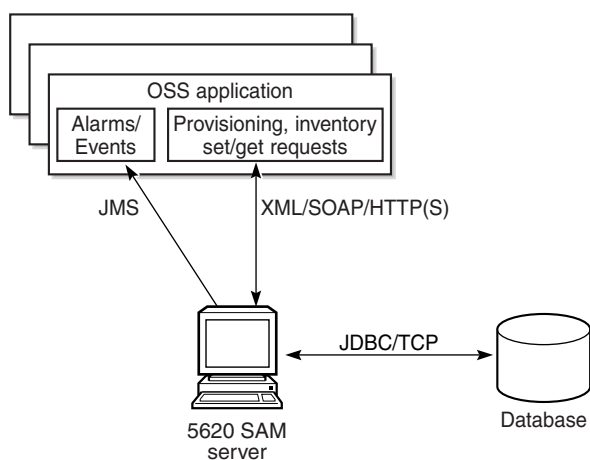
17901

3.3 Client interfaces

Client interfaces provide access to the 5620 SAM main server and to the managed network, as shown in Figure 3-2. The 5620 SAM-O clients can:

- send requests to the 5620 SAM main server to view and change objects, and to perform network operations
- monitor the 5620 SAM, and network events and alarms.

Figure 3-2 Client interfaces



21150

Event monitoring using JMS

Clients can monitor events in the 5620 SAM and the network by subscribing to event streams. This event-driven function allows a client to maintain a near-real-time view of events in the network. However, the function is unidirectional and does not allow clients to initiate changes in the 5620 SAM or the network.

To use JMS event streams to monitor events, the OSS developers need to create a JMS client. The client must subscribe to one or more event streams, which can be used to monitor events. Some applications are:

- monitoring alarms
- maintaining up-to-date inventory information by monitoring configuration changes in the network

See chapter 4 for more information about JMS and event monitoring.

Request and response using the 5620 SAM-O

Clients can request information and initiate changes in the 5620 SAM and in the managed network using the 5620 SAM-O API. This function is bidirectional. Clients send requests and receive responses that contain the information requested or the results of an operation.

The 5620 SAM-O API uses HTTP. The OSS developers must create a client that can send HTTP requests and receive responses. Some applications are:

- retrieving network inventory information
- changing the network configuration
- provisioning

The 5620 SAM-O API is also referred to as the XML API, although both the API and event monitoring use XML.

See chapter 5 for more information about using 5620 SAM-O API.

XML and SOAP

The JMS events and HTTP requests and responses are written in XML using the SOAP standard. XML is a standard format used to describe data. The 5620 SAM-O XML schema is defined in XML style sheets, which are distributed with the 5620 SAM-O documentation. API requests and responses, and JMS events are wrapped in SOAP envelopes.

For more information, see the related W3C specifications and other related material available from technical publishers and the Internet. Some suggested resources are:

- <http://www.w3.org/XML>
- <http://www.w3.org/TR/soap>
- Ray, Eric T. Learning XML (2nd Ed). O'Reilly, 2003

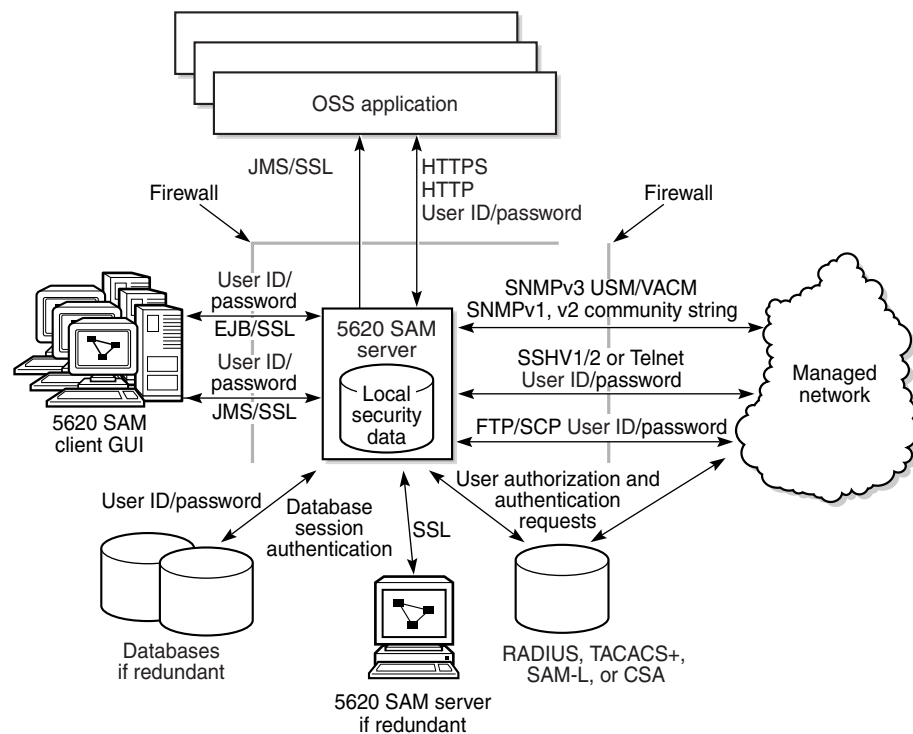
3.4 Secure communication

The 5620 SAM-O provides the following features for the secure communications between the 5620 SAM client OSS and the 5620 SAM-O server:

- HTTPS for XML requests and responses
- MD5-hashed passwords for requests
- SSL message encryption for JMS events

Figure 3-3 shows the 5620 SAM components and the available security mechanisms.

Figure 3-3 5620 SAM security



18083

SSL message encryption

5620 SAM-O clients can receive JMS XML messages securely by encrypting them through SSL. The SSL Certificate is configured on the 5620 SAM server. You must also configure the following client components:

- SSL keystore file on the OSS client
- Java to use the keystore file on the OSS client

See Procedure 4-3 and the Java user documentation for more information about configuring Java for SSL.

HTTPS communication with 5620 SAM-O

The 5620 SAM-O provides an optional web-based encryption protocol for sending requests and receiving responses between the 5620 SAM-O server and client. See the *5620 SAM User Guide* for information about how to configure the 5620 SAM server for HTTP or HTTPS.

Session management

Effective session management requires authentication, authorization, and accounting (AAA) functionality. Authentication is the verification of a user identification and password. Authorization is the assignment of different levels of access permissions to users. Accounting is the recording of user actions. A 5620 SAM operator can configure AAA functionality using the local security capability of the 5620 SAM server, a third-party authentication server, or a combination of local and third-party mechanisms.

- Local authentication on the 5620 SAM server is provided with a local database of users and a local security scheme to verify logon attempts and assign permission levels for command execution.
- Supported third-party authentication servers are RADIUS and TACACS+. These products run on their own platforms, with their own user lists and administration processes.

5620 SAM user accounts consist of a user name, password, priority, and an associated user group with scope of command and span of control profiles. User groups are used to assign and control user authorization levels, and to control the extent of access to such entities as customers, services, or faults. Users and groups can be assigned priorities that determine which user sessions take priority over others when system resources are limited. The system administrator can specify the maximum number of active user connections per user or user group. The system administrator can also limit the type of user access per managed device; for example, users might not be able to open a console.

A system administrator can use the 5620 SAM client GUI to manage 5620 SAM-O user account privileges, user sessions, SNMP format, user and group priority, and user authentication. See the *5620 SAM User Guide* for more information.

Client sessions

All 5620 SAM-O client sessions have username and password protection.

- Each OSS client XML/SOAP message is individually authenticated using cached information from an authorization server.
- JMS messages are protected by the user name and password for the JMS connection.

You can use an MD5-hashed or a clear text password to access the 5620 SAM server.

MD5-hashed password format

Alcatel-Lucent recommends that you send the password for a request in an MD5-hashed format. You can use the `md5hash password` utility to generate MD5-hashed passwords to access the 5620 SAM server. See Procedure 3-1 for more information about how to generate an MD5-hashed password for 5620 SAM server access.

Clear text password format

OSS users can access the 5620 SAM server using a clear text password. A clear text password provides no security for transmitted passwords unless HTTPS is used, but ensures greater compatibility with RADIUS and TACACS+ password storage. Clear text passwords should only be used if:

- the 5620 SAM-O uses an external mechanism for password authentication (such as RADIUS or TACACS)
- **and**
- a dedicated HTTPS channel is used for all XML/SOAP requests

Code 3-1 shows an example of the clear text password format.

Code 3-1: Sample clear text password format

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <security>
        <user>user name</user>
        <password hashed="false">password</password>
      </security>
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <ping xmlns="xmlapi_1.0"/>
  </SOAP:Body>
</SOAP:Envelope>
```

Procedure 3-1 To generate an MD5-hashed password for 5620 SAM main server access

- 1 Log on to a 5620 SAM main server station as the `samadmin` user.
- 2 Open a console window.
- 3 Enter the following to navigate to the server binary directory:

```
bash$ cd path/server/nms/bin ↵
```

where *path* is the main server installation directory, typically `/opt/5620sam/server`

- 4 Enter the following to generate an MD5-hashed password:

```
bash$ md5hash password string ↵
```

where *password string* is the list of words that you want to convert to a password in MD5-hashed format

For example, the MD5 hashed password for the word hello is the following:

```
5d41402abc4b2a76b9719d11017c592
```

- 5 Add the MD5-hashed password to the password field of the SOAP header for the specified user account.
-

5620 SAM-O security features controlled through the 5620 SAM client GUI

A system administrator can use the security management forms in the 5620 SAM client GUI to control the following security-related features for 5620 SAM-O:

- User account privileges
- Session monitoring
- Session shutdown
- SNMPv3 for secure NE polling
- Radius and TACAS+ authentication for 5620 SAM users
- Session logs of 5620 SAM client GUI and OSS activity. The 5620 SAM-O client ID must use the format for the JMS client ID described in section 4.5 to be uniquely identified as a 5620 SAM-O session.

You must create a scope of command profile to group the OSS Management scope of command role with additional roles. Otherwise, the 5620 SAM returns the following SOAP exception message:

```
<SOAP:Fault>
<faultcode>SOAP:Client</faultcode>
<faultstring>[security] Users require OSS Management privileges to
use SAM-O.</faultstring>
<faultactor>XmlApi</faultactor>
- <detail>
<requestID>XmlApiClient:0</requestID>
</detail>
</SOAP:Fault>
```

See the *5620 SAM User Guide* for information about managing security on 5620 SAM user groups and the associated scope of command and span of control profiles.

If a 5620 SAM-O user supplies an incorrect username or password, or if the username or password is missing, the 5620 SAM returns the following SOAP exception message:

```
<SOAP:Fault>
<faultcode>SOAP:Client</faultcode>
<faultstring>[security] Login failure.</faultstring>
```

```
<faultactor>XmlApi</faultactor>
- <detail>
<requestID>XmlApiClient:0</requestID>
</detail>
</SOAP:Fault>
```

If a 5620 SAM-O user is not assigned the appropriate scope of command privilege for a class type, the following exception is returned:

```
[ app: generic ] [ class: GenericObject ] [ instance: instance:object ]
[ cause: Method Invocation Failed (13) ]
[ descr: Insufficient privileges to perform this operation.
User does not have create-update-delete access to class classname ]
[ nested exception: null ]
```

If a 5620 SAM-O user is not assigned the appropriate span of control for an object, the following exception is returned:

```
[ app: generic ] [ class: generic.GenericObject ] [ instance:
-unknown- ] [ descr: SecurityManager: Authorization Failure:
User does not have access to objectFullName. Object is not in user's span.
[exceptionClass: SecurityException]
```

See the *5620 SAM User Guide* for information about how listing is filtered by span, and for how configuration is span enforced.

Unlike in HTTP or HTTPS where span of control rules are implicitly enforced, span of control rules are not implicitly enforced via JMS. JMS requires the administrator to explicitly specify the required span IDs within the JMS filter. For example, without the correct JMS filter, it may not be possible to listen to events from objects that are not in the 5620 SAM-O user's span of control. Table 3-1 lists example JMS filter entries.

Table 3-1 Example JMS filter entries

JMS filter	Description
ALA_span like '%:2:%'	A JMS query that uses this filter results in JMS messages being sent from objects in span 2 (for example, Default Router Span). Events from objects in other spans are not published.
ALA_span like '%:21:%' or ALA_span like '%:0:%'	A JMS query that uses this filter results in JMS messages being sent from objects in span 21 or non-span objects. Events from objects in other spans are not published.
(ALA_span like '%:1:%' or ALA_span like '%:2:%' or ALA_span like '%:3:%' or ALA_span like '%:4:%' or ALA_span like '%:5:%' or ALA_span like '%:6:%' or ALA_span like '%:7:%' or ALA_span like '%:8:%' or ALA_span like '%:0:%') and (ALA_span not like '%:22:%')	A JMS query that uses this filter results in JMS messages being sent from all objects (all of Default Spans listed) except those blocked in span 22.

3.5 5620 SAM-O set up and operation

Use the following procedure to set up and operate the 5620 SAM-O.

Procedure 3-2 To set up and operate the 5620 SAM-O

- 1 Ensure that the 5620 SAM server is installed and that the license key supports the use of the 5620 SAM-O software.
- 2 Develop an understanding of the information model and the 5620 SAM-O XML message structure. See chapter 6.
- 3 Determine whether the OSS application is request-response driven, or acts as a passive event listener (many OSS applications do both). See section 3.3 for an overview of both approaches.



Note — Training and developer support are available from Alcatel-Lucent. Contact your Alcatel-Lucent account or OIPS technical support representative for more information.

- 4 If you are communicating with the 5620 SAM-O connection using JMS, see chapter 4.

If you are communicating with the 5620 SAM-O connection using XML, see chapter 5.

4 — *Event monitoring using JMS*

- 4.1 Overview 4-2
- 4.2 JMS client planning considerations 4-2
- 4.3 JMS connections 4-2
- 4.4 Workflow to set up and operate a JMS connection 4-3
- 4.5 JMS subscriptions 4-4
- 4.6 JMS events 4-6
- 4.7 JMS message filtering 4-23
- 4.8 JMS and redundancy 4-36
- 4.9 Connection monitoring and error recovery 4-37
- 4.10 Managing and monitoring JMS sessions 4-42
- 4.11 JMS consumer creation 4-43
- 4.12 JMS record and playback tool 4-46

4.1 Overview

The 5620 SAM-O clients use JMS channels to receive real-time network event information from the server. The event types that 5620 SAM-O clients receive include:

- managed network alarm notifications
- managed network configuration changes
- server activity-switch notifications
- 5620 SAM database connectivity errors

A JMS consumer is a JMS client that receives events from a JMS provider. In the 5620 SAM-O, the 5620 SAM is the JMS provider and the OSS is the consumer. An OSS developer must write a JMS client that connects to the 5620 SAM, subscribes to the topic or topics that are relevant to the OSS, and processes events. See section 4.11 for information about creating a JMS consumer.

The following sections provide the information necessary to create a JMS client that communicates with the 5620 SAM, in addition to sample code for a JMS client. You must have a thorough understanding of JMS to monitor 5620 SAM-O events and understand the topics discussed in this chapter. JMS is a published industry standard and is beyond the scope of the *5620 SAM XML OSS Interface Developer Guide*. See <http://www.oracle.com/technetwork/java> for more information.

4.2 JMS client planning considerations

Consider the following when planning and developing your JMS client:

- A JMS client must use the same Java version, or later, as the 5620 SAM. See the *5620 SAM Planning Guide* to determine the supported version.
- A JMS client connecting to the 5620 SAM-O needs to have the samOss.jar in the classpath. This file is located at:
`http://main_server_address:8080/integration/samOss.jar`. See Procedure 4-3 for more information about the samOss.jar file.

4.3 JMS connections

An OSS application monitors events by establishing a JMS connection to the 5620 SAM, then by creating a subscription to one of the available JMS topics.



Note — The 5620 SAM supports a maximum of 10 JMS connections at a time. Since these resources are shared, Alcatel-Lucent strongly recommends that an OSS use only one connection. If an OSS uses more than one connection, Alcatel-Lucent requires that the number of connections be configurable at runtime, that one connection be a valid option, and that the number of connections used and how to configure this be clearly documented.

You require the following information to establish a JMS connection to the 5620 SAM:

- a server address and port
- a 5620 SAM-O user name and password

The server address can be the IP address of the server or the server hostname. The default port for 5620 SAM standalone installations is port 1099. The default port is 1100 for redundant installations. See section 4.8 for information about how to connect to a redundant system. The 5620 SAM server ports and addresses must be configurable in each client OSS application. If a firewall exists between the 5620 SAM server and client OSS application, multiple ports must be opened in the firewall. See the *5620 SAM Planning Guide* for the list of ports required for JMS and other connections.

When you create a connection, you require a valid 5620 SAM username and password. The password can be MD5-hashed or in plain text. The 5620 SAM user must use a scope of command profile that includes the OSS role. See the *5620 SAM User Guide* for more information about how to manage 5620 SAM users and profiles.

4.4 Workflow to set up and operate a JMS connection

The following workflow lists the high-level steps required to set up and operate a JMS connection. This workflow assumes that you have completed Procedure 3-2 to set up and operate the 5620 SAM-O.

- 1 Develop an understanding of the 5620 SAM-O JMS message interface. See section 4.1 for more information.
- 2 Determine the events that your application must monitor. See section 4.7 for more information about defining a message filter to include the specific messages required by the OSS application.
- 3 Determine the topic or topics that your application must monitor to receive the required events. See Table 4-1 for more information about the available topics.
- 4 Develop a JMS client. See Procedure 4-3.
- 5 Open a session to the 5620 SAM-O server and monitor events and exceptions.
- 6 Subscribe to topics that are of interest to the application.
- 7 Manage the connection and any errors related to communication failure.
- 8 As required, resync the OSS database with the 5620 SAM. See Procedure 4-1.
- 9 As required, remove the durable subscription when disconnecting from the 5620 SAM. See Procedure 4-2.

4.5 JMS subscriptions

After you establish a JMS connection, you require the following to create a new subscription:

- a client ID
- a JMS topic
- a filter
- an acknowledgement mode
- a persistence mode



Note — Alcatel-Lucent strongly recommends that an OSS establishes only one JMS subscription for each JMS connection.

Client IDs

You must establish a client ID for the JMS client before you create a JMS subscription. The client ID is used to identify a client in the JMS and 5620 SAM logs, and in the 5620 SAM GUI. In addition, the client ID is used to send JMS events to specific subscribers, such as missed events—when only a single subscriber is affected—or file available events for files requested by a specific client. The client ID must also be used in the subscription filter.

The 5620 SAM supports the following format for the client ID:

`<unique_name>@<numeric_id>`

The `<unique_name>` must be a string that identifies the client to which the subscription belongs, usually the name of the OSS product. All of the subscriptions from the same OSS must use the same `<unique_name>`. The `<numeric_ID>` is used to differentiate between different subscriptions that belong to the same OSS.

JMS topics

A JMS client controls which event messages it receives by subscribing to the appropriate JMS topic and filtering the events. A JMS subscription must include a filter that defines the types of events the client is to receive. Table 4-1 lists the available 5620 SAM JMS topics.

Table 4-1 JMS topics

Topic	Description
5620-SAM-topic-xml	Subscribe to all events, such as object creation and deletion. This topic publishes all events with no dependency on other topics.
5620-SAM-topic-xml-filtered	Subscribe to all events using a filter to limit the number of JMS messages that are sent.

(1 of 2)

Topic	Description
5620-SAM-topic-xml-general	Subscribe to general events, such as object creation and deletion. This topic contains objects that do not fall into the fault, file, or stats topics.
5620-SAM-topic-xml-fault	Subscribe to fault events.
5620-SAM-topic-xml-file	Subscribe to findToFile events.
5620-SAM-topic-xml-stats	Subscribe to events related to statistics collection.

(2 of 2)

The 5620-SAM-topic-xml sends all events. You can use other topics to receive specific events. For example, the 5620-SAM-topic-xml-fault topic sends only fault events.

The 5620-SAM-topic-xml-filtered topic allows the filtering of events before they are sent, based on the properties of the objects associated with the events. This reduces the amount of JMS event traffic between the 5620 SAM and OSS clients. See section 4.7 for more information about using filters to specify which events JMS sends.

Filters

A subscription to a JMS topic must include a filter definition that specifies which events are to be sent to the client. If you subscribe to the 5620-SAM-topic-xml-filtered topic, you can use advanced filters to further restrict the returned events based on properties of the objects associated with events. See section 4.7 for more information about simple and advanced JMS event filtering.

Acknowledgement modes

Clients can specify an acknowledgement mode for JMS sessions. The following acknowledgement modes are supported:

- **AUTO_ACKNOWLEDGE**

The JMS client automatically acknowledges each message received from the server and another message is not sent by the server until an acknowledgement is received from the JMS client. This mode does not efficiently use the available bandwidth, and performance degrades significantly as latency increases.

- **DUPS_OK_ACKNOWLEDGE**

This mode allows new messages to be sent to the JMS client before earlier messages are acknowledged. This mode allows for a higher throughput for messages in high-latency networks. In addition, this mode reduces session overhead and per message delay by minimizing the work the session does to prevent duplicates. Available bandwidth to the client is used more efficiently and performance does not degrade significantly as latency increases.

Alcatel-Lucent recommends that you use the DUPS_OK_ACKNOWLEDGE mode because of the improvements in performance. However, consider the following when you implement the JMS consumer. If there is a connectivity failure, the 5620 SAM re-sends all unacknowledged messages once connectivity is restored. When you use the AUTO_ACKNOWLEDGE mode, this can result in the client receiving a single

duplicate message—if that message was received by the client but the connectivity failure prevented the 5620 SAM from receiving the acknowledgement. When you use the DUPS_OK_ACKNOWLEDGE mode, there may be several messages at any time that have been sent by the 5620 SAM for which the 5620 SAM has not received an acknowledgement. All of these messages are resent once connectivity is restored.

Code 4-1 shows DUPS_OK__ACKNOWLEDGE as the message acknowledgement mode.

Code 4-1: JMS client message acknowledgement mode

```
topicSession =  
topicConnection.createTopicSession(false, TopicSession.DUPS_OK_ACKNOWLEDGE );
```

Persistence modes

There are two persistence modes available to subscribers—durable and non-durable. Durable subscriptions can continue to exist even if the client is not connected. If the client is not connected, the 5620 SAM queues messages until the client reconnects. Non-durable subscriptions are removed if the client is disconnected, and messages are not queued.

The tolerance of the OSS for missed events determines whether it is best to use durable or non-durable subscriptions. If there is a loss of connectivity, non-durable subscriptions lose events and may also lose events silently because of an internal server queue overflow. For durable subscriptions, events are queued if there is a connectivity failure, and are available when connectivity is restored.

For both durable and non-durable subscriptions, if an internal queue overflow results in a missed event, the subscriber is sent a notification of missed events, which allows the subscriber to take corrective action. See “Missed events” in section 4.9 for more information about missed events.

4.6 JMS events

An OSS client subscribes to a JMS topic to listen to events. The following subsections describe typical events, the event formats, and a list of the available event classes.

JMS XML schemas

JMS XML schemas describe the structure and allowed elements of a document, similar to a DTD.

The 5620 SAM documentation contains information about the 5620 SAM-O schemas, objects, methods, parameters, and changes between releases.

Table 4-2 lists files that contain XML schema information related to JMS.

Table 4-2 JMS XML schema files

File	Description
xmlApiJms.xsd	JMS general messages
xmlApiJmsHeader.xsd	JMS messages for filtering
xmlApiJmsTypes.xsd	Types used in the JMS header and body
xmlApiJmsBody.xsd	Event body format

Some events contain complete objects defined in the standard 5620 SAM XML schemas. See “[JMS event classes](#)” in this section for more information. See chapter 3 for information about the W3C standards for XML.

JMS event header properties

The following tables list and describe the properties in JMS event message headers. Table 4-3 lists the properties that are included in all events. Table 4-4 lists properties that are included in some alarm-related events.

Table 4-3 Event header properties—all events

Property	Values	Description
ALA_allomorphic	Allomorphic class	The allomorphic class of the object type, if applicable
ALA_category	ACCOUNTING—accounting statistics CPAM—events related to 5650 CPAM topology management DATABASE—database state messages EQUIPMENT—physical equipment events FAULT—fault management events (alarms) GENERAL—events that are not in another category STATISTICS—statistics, except accounting statistics SERVICE—service-related events SOFTWARE—not currently used	The event category
ALA_clientId	OSS client ID, or empty	The client ID associated with the event, if applicable
ALA_isVessel	false true	Whether the message is an event vessel
ALA_OLC	0 (object has no OLC state) 1 (maintenance) 2 (in service)	OLC state. Applies to the creation, attribute changes, and deletion for objects that have an OLC state. See the <i>5620 SAM User Guide</i> for more information about OLC states.
ALA_routeManager	Application name of the route manager	Present if the message contains a managed-route change
ALA_span	List of control IDs	The OSS user span of control ID list
ALA_topic	JMS topic name	The JMS topic of the message

(1 of 2)

Property	Values	Description
ALA_vesselSize	0 to 250	Size, in messages, of the event vessel; present only if ALA_isVessel value is true
eventName	See Table 4-5 for a list of the supported objects.	The JMS event class name, for example, ObjectDeletion for a deletion event.
MTOSI_NTType	ALA_OTHER NT_OBJECTDELETION NT_OBJECTCREATION NT_ATTRIBUTE_VALUE_CHANGE NT_STATE_CHANGE NT_ALARM NT_HEARTBEAT ALA_RELATIONSHIPCHANGE	The type of notification
MTOSI_objectName	Object name	The name of the associated object
MTOSI_objectType	KeepAliveEvent AlarmStatusChangeEvent DBActivityEvent DBProxyStateChangeEvent DBErrorEvent DBConnectionStateChangeEvent StateChangeEvent FileAvailableEvent DeployerEvent TerminateClientSession Any network object names in the form of <i>packageName.ClassName</i> for the following events: <ul style="list-style-type: none"> • AttributeValueChangeEvent • ObjectCreationEvent • ObjectDeletionEvent • RelationshipChangeEvent • StatsEvent 	The type of associated object
MTOSI_osTime	Time, in ms, since January 1, 1970	The server system time at event creation

(2 of 2)

Table 4-4 Event header properties—alarm events

Property	Values	Description
ALA_alarmType	Alarm type	The type of alarm
ALA_isCorr	True, if alarm is correlated	Whether the alarm is correlated, that is, caused by another alarm. See the <i>5620 SAM User Guide</i> for information about correlated alarms.
MTOSI_aliasNameList	Alarm name	The alarm name
MTOSI_perceivedSeverity	Severity	The alarm severity
MTOSI_probableCause	Probable cause	The probable cause associated with the alarm
MTOSI_serviceAffecting	True if service-affecting	Whether the alarm is service-affecting

JMS event classes

Table 4-5 lists the JMS event classes. Each highlighted class name in the table is a link to sample event output.

Table 4-5 JMS event classes

Event type	Description
AlarmStatusChangeEvent	Indicates an alarm status change
AttributeValueChangeEvent	Indicates a change to one or more parameter in the database
DBActivityEvent	Indicates database switchover or failover activity
DBConnectionStateChangeEvent	Indicates a change in the database connection state
DBErrorEvent	Indicates a new database error or the correction of an earlier error
DBProxyStateChangeEvent	Indicates a database proxy status change
DeployerEvent	Indicates the success or failure of an asynchronous deployment request; is available only in XML topics
EventVessel	Indicates that the message is an event vessel that contains one or more events
ExceptionEventXMLFormat	Indicates the occurrence of an exception
FileAvailableEvent	Indicates that a file is available for retrieval; applies to asynchronous and synchronous findToFile requests
IncrementalRequestEvent	Generated for activity related to generic.GenericObject.triggerIncrementalRequest
KeepAliveEvent	A keep-alive message that is sent periodically to ensure that the server is running and can communicate with the OSS client
LogFileAvailableEvent	Indicates that an accounting statistics file is ready for retrieval
ManagedRouteEvent	Indicates a change to the routing topology managed by the 5650 CPAM
ObjectCreationEvent	Identifies a newly created object; contains the object details
ObjectDeletionEvent	Identifies a newly deleted object; contains the object details
RelationshipChangeEvent	Indicates that a relationship has changed between objects
ScriptExecutionEvent	Indicates script execution by an OSS or GUI client. If specified, the message is sent only to the client that executed the script. Identifies script execution success or failure, the result full name, and the location of the file that contains the result.
StateChangeEvent	Indicates a server change, such as reloaded alarm information or changed alarm count. An example of an important state change event is the SystemInfoEvent, which is sent each time an OSS client creates a new subscription or connects to a subscription.
StatsEvent	Indicates statistics-collection activity; includes the statistics type, collection time, and the NE identifier, and marks the start or end of polling for an NE, as indicated by the <state> parameter
TerminateClientSessionEvent	Identifies a client session to close for security reasons
XMLFilterChangeEvent	Indicates a successful XML filter change, as requested by the OSS client

AlarmStatusChangeEvent

Code 4-2 shows sample event output for the AlarmStatusChangeEvent class.

Code 4-2: Sample AlarmStatusChangeEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138396703631</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>AlarmStatusChange</MTOSI_objectType>
      <MTOSI_NTType>NT_ATTRIBUTE_VALUE_CHANGE</MTOSI_NTType>
      <ALA_category>FAULT</ALA_category>
      <ALA_allomorphic>TiEvent</ALA_allomorphic>
      <MTOSI_objectName>svc-mgr:service-3:10.1.1.143</MTOSI_objectName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <alarmStatusChangeEvent>
        <objectFullName>svc-mgr:service-3:10.1.1.143</objectFullName>
        <attribute>
          <attributeName>alarmStatus</attributeName>
          <value>2</value>
        </attribute>
      </alarmStatusChangeEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

AttributeValueChangeEvent

Code 4-3 shows sample event output for the AttributeValueChangeEvent class. An AttributeValueChangeEvent consists of a pointer with a list of changed attributes. For each attribute, both the old value and the new value are included in the message.

For more information about the attributes, or parameters, of specific classes, see the *5620 SAM-O XML Reference*. For parameters that are enumerators, the integer value is used for AttributeValueChangeEvents, while the string value is used in ObjectCreationEvents.

Code 4-3: Sample AttributeValueChange output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138123820148</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>netw.NetworkElement</MTOSI_objectType>
      <MTOSI_NTType>NT_ATTRIBUTE_VALUE_CHANGE</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_allomorphic>netw.NetworkElement</ALA_allomorphic>
      <MTOSI_objectName>network:10.1.186.218</MTOSI_objectName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <attributeValueChangeEvent>
        <objectFullName>network:10.1.186.218</objectFullName>
        <attribute>
          <attributeName>resyncStatus</attributeName>
          <newValue>
            <int>2</int>
          </newValue>
          <oldValue>
            <int>3</int>
          </oldValue>
        </attribute>
      </attributeValueChangeEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```



```

        <attributeName>lastTimeResyncStarted</attributeName>
        <newValue>
            <long>1138123819632</long>
        </newValue>
        <oldValue>
            <long>1138123519629</long>
        </oldValue>
    </attribute>
</attributeValueChangeEvent>
</jms>
</SOAP:Body>
</SOAP:Envelope>

```

DBActivityEvent

Code 4-4 shows sample event output for the DBActivityEvent class.

Code 4-4: Sample DBActivityEvent output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138395709210</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>DBActivityEvent</MTOSI_objectType>
      <MTOSI_NTType>NT_STATE_CHANGE</MTOSI_NTType>
      <ALA_category>DATABASE</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <dbActivityEvent>
        <state>failoverEnd</state>
      </dbActivityEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

DBConnectionStateChangeEvent

Code 4-5 shows sample event output for the DBConnectionStateChangeEvent class.

Code 4-5: Sample DBConnectionStateChangeEvent output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138395709210</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>DBConnectionStateChangeEvent</MTOSI_objectType>
      <MTOSI_NTType>NT_STATE_CHANGE</MTOSI_NTType>
      <ALA_category>DATABASE</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <dbConnectionStateChangeEvent>
        <state>connectionUp</state>
      </dbConnectionStateChangeEvent>
    </jms>
  </SOAP:Body>

```

```
</SOAP:Envelope>
```

DBErrorEvent

Code 4-6 shows sample event output for the DBErrorEvent class.

Code 4-6: Sample DBErrorEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138395709210</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>DBErrorEvent</MTOSI_objectType>
      <MTOSI_NTType>NT_STATE_CHANGE</MTOSI_NTType>
      <ALA_category>DATABASE</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <dbErrorEvent>
        <state>clear</state>
        <error />
      </dbErrorEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

DBProxyStateChangeEvent

Code 4-7 shows sample event output for the DBProxyStateChangeEvent class.

Code 4-7: Sample DBProxyStateChangeEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138395709210</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>DBProxyStateChangeEvent</MTOSI_objectType>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_category>DATABASE</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <dbProxyStateChangeEvent>
        <state>databaseProxyPrimaryUp</state>
      </dbProxyStateChangeEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

DeployerEvent

Code 4-8 shows sample event output for the DeployerEvent class.

Code 4-8: Sample DeployerEvent output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1165863370530</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>DeployerEvent</MTOSI_objectType>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <deployerEvent>
        <requestId>XmlApiClient:3</requestId>
        <successList>
          <deployerId>Default.DeployerBank:depl-486</deployerId>
        </successList>
        <failedList>
          <deployerId>Default.DeployerBank:depl-487</deployerId>
        </failedList>
      </deployerEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

EventVessel

Code 4-9 shows sample event output for the EventVessel class.

Code 4-9: Sample EventVessel output

```

<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>EventVessel</eventName>
      <ALA_category>GENERAL</ALA_category>
      <ALA_vesselSize>3</ALA_vesselSize>
      <ALA_classNameList>fm.AlarmObject,lte.SubscAndEquipmentTrac
es,lte.ENBNESpecifics</ALA_classNameList>
      <ALA_isVessel>true</ALA_isVessel>
      <MTOSI_osTime>1308318280367</MTOSI_osTime>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_alamorphicClassNameList>fm.AlarmObject,lte.SubscAndEqu
ipmentTraces,lte.ENBNESpecifics</ALA_alamorphicClassNameList>
      <ALA_clientId>JMSTEST@1</ALA_clientId>
      <ALA_eventName>EventVessel</ALA_eventName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <eventVessel>
        <attributeValueChangeEvent>
          <objectFullName>faultManager:network@192.168.101.13|a
larm-31-4-24</objectFullName>
          <attribute>

```

```

        <attributeName>numberOfOccurencesSinceClear</attributeName>
        <newValue>
            <long>3</long>
        </newValue>
        <oldValue>
            <long>2</long>
        </oldValue>
    </attribute>
</attributeValueChangeEvent>
<statsEvent>
    <state>begin</state>
    <networkElement>192.168.101.13</networkElement>
    <statsType>accounting</statsType>
    <time>1308256029287</time>
</statsEvent>
<statsEvent>
    <state>end</state>
    <networkElement>192.168.101.13</networkElement>
    <statsType>accounting</statsType>
    <time>1308256029302</time>
</statsEvent>
</eventVessel>
</jms>
</SOAP:Body>
</SOAP:Envelope>
```

ExceptionEventXMLFormat

Code 4-10 shows sample event output for the ExceptionEventXMLFormat class.

Code 4-10: Sample ExceptionEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1174926836987</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>aengr.Policy</MTOSI_objectType>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <ExceptionEventXMLFormat>
        <className>aengr.Policy</className>
        <operation>distribute on node 10.168.1.91</operation>
        <objectName>Access Egress:2100</objectName>
        <requestId>client1:0</requestId>
        <errorMessage>[ app: autoconfigChild ] [ class: aengr.Policy ] [
instance: network:10.168.1.91:Access Egress:2100 ] [ descr: invalid action
bitmask: 3 : Object deletion is in progress, the object cannot be configured
:Parent = network:10.168.1.91:Access Egress:2100: child = queue-1
]</errorMessage>
      </ExceptionEventXMLFormat>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

FileAvailableEvent

Code 4-11 shows sample event output for the FileAvailableEvent class.

Code 4-11: Sample FileAvailableEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138129016027</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>FileAvailableEvent</MTOSI_objectType>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <fileAvailableEvent>
        <fileName>equipmentPhysicalPort.xml</fileName>
      </fileAvailableEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

IncrementalRequestEvent

Code 4-12 shows sample event output for the IncrementalRequestEvent class.

Code 4-12: Sample IncrementalRequestEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>IncrementalRequestEvent</eventName>
      <ALA_category>GENERAL</ALA_category>
      <ALA_OLC>0</ALA_OLC>
      <ALA_isVessel>false</ALA_isVessel>
      <ALA_allomorphic></ALA_allomorphic>
      <MTOSI_osTime>1276545012449</MTOSI_osTime>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <MTOSI_objectName></MTOSI_objectName>
      <ALA_clientId></ALA_clientId>
      <MTOSI_objectType>IncrementalRequestEvent</MTOSI_objectType>
      <ALA_eventName>IncrementalRequestEvent</ALA_eventName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <IncrementalRequestEvent>
        <requestId>4</requestId>
        <incrementalAction>RE_EVAL_MSAPS_ON_MSAP_POLICY</incrementalActi
on>
        <serviceType>default</serviceType>
        <status>finished</status>
        <originalParam>
          <string>msapPolicy:msappolicy</string>
        </originalParam>
        <payload>
          <boolean>true</boolean>
        </payload>
      </IncrementalRequestEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

KeepAliveEvent

Code 4-13 shows sample event output for the KeepAliveEvent class.

Code 4-13: Sample KeepAliveEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1137166251642</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>KeepAliveEvent</MTOSI_objectType>
      <MTOSI_NTType>NT_HEARTBEAT</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <keepAliveEvent />
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

LogFileAvailableEvent

Code 4-14 shows sample event output for the LogFileAvailableEvent class.

Code 4-14: Sample LogFileAvailableEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>LogFileAvailable</eventName>
      <MTOSI_osTime>1184770068031</MTOSI_osTime>
      <ALA_clientId>JMS_Client_ID</ALA_clientId>
      <MTOSI_objectType>LogFileAvailableEvent</MTOSI_objectType>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <logfileAvailableEvent>
        <fileName>client_directory\NE_IP_address_1184770068031.xml</file
Name>
        <serverIpAddress>server_IP_address</serverIpAddress>
        <routerId>NE_IP_address</routerId>
      </logfileAvailableEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

ManagedRouteEvent

Code 4-15 shows sample event output for the ManagedRouteEvent class.

Code 4-15: Sample ManagedRouteEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
```

```
<header xmlns="xamlapi_1.0">
  <eventName>ManagedRouteEvent</eventName>
  <ALA_category>CPAM</ALA_category>
  <ALA_OLC>0</ALA_OLC>
  <ALA_application>1</ALA_application>
  <ALA_routeManager>tpgy-mgr:application-diane</ALA_routeManager>
  <ALA_isVessel>false</ALA_isVessel>
  <ALA_allomorphic>Hermes</ALA_allomorphic>
  <MTOSI_osTime>1250787917539</MTOSI_osTime>
  <MTOSI_NTType>ALA_MANAGEDROUTE</MTOSI_NTType>
  <MTOSI_objectName/>
  <ALA_clientId/>
  <MTOSI_objectType>ManagedRouteEvent</MTOSI_objectType>
  <ALA_eventName>ManagedRouteEvent</ALA_eventName>
</header>
</SOAP:Header>
<SOAP:Body>
  <jms xmlns="xamlapi_1.0">
    <managedRouteEvent>
      <item>
        <key>
          <topology.RouteKey>
            <sourceType>ipv4</sourceType>
            <source>192.168.202.175</source>
            <sourceLen>32</sourceLen>
            <destType>ipv4</destType>
            <dest>192.168.202.171</dest>
            <destLen>32</destLen>
          </topology.RouteKey>
        </key>
        <value>
          <topology.RouteResult>
            <errorCode>mospfCouldNotDeterminAs</errorCode>
            <errorMessage>Could not determine administrative
domain.</errorMessage>
            <type>standard</type>
            <vertices/>
            <detailedErrors>
              <topology.DetailedError>
                <errorCode>mospfCouldNotDeterminAs</errorCode>
                <message>Could not determine administrative
domain.</message>
              </topology.DetailedError>
            </detailedErrors>
          </topology.RouteResult>
        </value>
      </item>
    </managedRouteEvent>
  </jms>
</SOAP:Body>
</SOAP:Envelope>
```

ObjectCreationEvent

Code 4-16 shows sample event output for the `ObjectCreationEvent` class.

Code 4-16: Sample ObjectCreationEvent output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xamlapi_1.0">
      <MTOSI_osTime>1138128165344</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>aclfilter.MacFilterEntry</MTOSI_objectType>
      <MTOSI_NTType>NT_OBJECTCREATION</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_allomorphic>aclfilter.MacFilterEntry</ALA_allomorphic>
    </header>
  </SOAP:Header>
  <body>
    <MTOSI_createMacFilterEntry />
  </body>
</SOAP:Envelope>
```

```

        <MTOSI_objectName>MAC Filter:3:entry-1</MTOSI_objectName>
    </header>
</SOAP:Header>
<SOAP:Body>
    <jms xmlns="xmlapi_1.0">
        <objectCreationEvent>
            <aclfilter.MacFilterEntry>
                <frameType>unspecified</frameType>
                <sourceMacAddress>10-00-00-00-00-10</sourceMacAddress>
                <sourceMacAddressMask>00-00-00-00-00-00</sourceMacAddressMask>
            >
                <destinationMacAddress>10-00-00-00-00-10</destinationMacAddress>
            ss>
                <destinationMacAddressMask>00-00-00-00-00-00</destinationMacAddressMask>
            ddressMask>
                <dot1pValue>notSet</dot1pValue>
                <dot1pMask>unspecified</dot1pMask>
                <ethernetType>-1</ethernetType>
                <dsap>-1</dsap>
                <dsapMask>-1</dsapMask>
                <ssap>-1</ssap>
                <ssapMask>-1</ssapMask>
                <snapPid>-1</snapPid>
                <snapOui>off</snapOui>
                <action>default</action>
                <logId>0</logId>
                <administrativeState>notInService</administrativeState>
                <containingPolicyDisplayedName>ACL Mac Filter
2</containingPolicyDisplayedName>
                <containingPolicyId>3</containingPolicyId>
                <policyType>macAcl</policyType>
                <isLocal>>false</isLocal>
                <siteId>0.0.0.0</siteId>
                <siteName>N/A</siteName>
                <displayedName>Filter 10-00-00-00-00-10</displayedName>
                <description>Filter Entry # 1</description>
                <id>1</id>
                <globalPolicy>N/A</globalPolicy>
                <deploymentState>0</deploymentState>
                <objectFullName>MAC Filter:3:entry-1</objectFullName>
                <name>Filter 10-00-00-00-00-10</name>
                <selfAlarmed>>false</selfAlarmed>
                <children-Set />
            </aclfilter.MacFilterEntry>
        </objectCreationEvent>
    </jms>
</SOAP:Body>
</SOAP:Envelope>

```

ObjectDeletionEvent

Code 4-17 shows sample event output for the ObjectDeletionEvent class.

Code 4-17: Sample ObjectDeletionEvent output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Header>
        <header xmlns="xmlapi_1.0">
            <eventName>ObjectDeletion</eventName>
            <MTOSI_osTime>1224518670453</MTOSI_osTime>
            <ALA_clientId/>
            <MTOSI_NTType>NT_OBJECTDELETION</MTOSI_NTType>
            <MTOSI_objectType>equipment.PhysicalPort</MTOSI_objectType>
            <ALA_category>EQUIPMENT</ALA_category>
            <ALA_isVessel>>false</ALA_isVessel>
            <ALA_allomorphic>equipment.PhysicalPort</ALA_allomorphic>
            <ALA_eventName>ObjectDeletion</ALA_eventName>
            <ALA_span>:0:</ALA_span>
        </header>
    </SOAP:Header>
    <SOAP:Body>
        <ObjectDeletionEvent>
            <objectName>MAC Filter:3:entry-1</objectName>
            <sourceMacAddress>10-00-00-00-00-10</sourceMacAddress>
            <sourceMacAddressMask>00-00-00-00-00-00</sourceMacAddressMask>
            <destinationMacAddress>10-00-00-00-00-10</destinationMacAddress>
            <destinationMacAddressMask>00-00-00-00-00-00</destinationMacAddressMask>
            <dot1pValue>notSet</dot1pValue>
            <dot1pMask>unspecified</dot1pMask>
            <ethernetType>-1</ethernetType>
            <dsap>-1</dsap>
            <dsapMask>-1</dsapMask>
            <ssap>-1</ssap>
            <ssapMask>-1</ssapMask>
            <snapPid>-1</snapPid>
            <snapOui>off</snapOui>
            <action>default</action>
            <logId>0</logId>
            <administrativeState>notInService</administrativeState>
            <containingPolicyDisplayedName>ACL Mac Filter
            <containingPolicyId>3</containingPolicyId>
            <policyType>macAcl</policyType>
            <isLocal>>false</isLocal>
            <siteId>0.0.0.0</siteId>
            <siteName>N/A</siteName>
            <displayedName>Filter 10-00-00-00-00-10</displayedName>
            <description>Filter Entry # 1</description>
            <id>1</id>
            <globalPolicy>N/A</globalPolicy>
            <deploymentState>0</deploymentState>
            <objectFullName>MAC Filter:3:entry-1</objectFullName>
            <name>Filter 10-00-00-00-00-10</name>
            <selfAlarmed>>false</selfAlarmed>
            <children-Set />
        </ObjectDeletionEvent>
    </SOAP:Body>
</SOAP:Envelope>

```



```

        <MTOSI_objectName>network:15.1.1.89:shelf-1:cardSlot-1:card:daughte
rCardSlot-1:daughterCard:port-53</MTOSI_objectName>
        <ALA_OLC>2</ALA_OLC>
    </header>
</SOAP:Header>
<SOAP:Body>
    <jms xmlns="xmlapi_1.0">
        <objectDeletionEvent>
            <objectFullName>network:15.1.1.89:shelf-1:cardSlot-1:card:daught
erCardSlot-1:daughterCard:port-53</objectFullName>
            </objectDeletionEvent>
        </jms>
    </SOAP:Body>
</SOAP:Envelope>

```

RelationshipChangeEvent

Code 4-18 shows sample event output for the RelationshipChangeEvent class.

Code 4-18: Sample RelationshipChangeEvent output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138386054650</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>epipe.Epipe</MTOSI_objectType>
      <MTOSI_NTType>ALA_RELATIONSHIPCHANGE</MTOSI_NTType>
      <ALA_category>SERVICE</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName>svc-mgr:service-3</MTOSI_objectName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <relationshipChangeEvent>
        <objectFullName>svc-mgr:service-3</objectFullName>
        <relationship>
          <changeType>added</changeType>
          <fromObjectName>svc-mgr:service-3</fromObjectName>
          <fromObjectClass>epipe.Epipe</fromObjectClass>
          <toObjectName>subscriber:1</toObjectName>
          <toObjectClass>subscr.Subscriber</toObjectClass>
        </relationship>
      </relationshipChangeEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

ScriptExecutionEvent

Code 4-19 shows sample event output for the ScriptExecutionEvent class.

Code 4-19: Sample ScriptExecutionEvent output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1173717919050</MTOSI_osTime>
      <ALA_clientId>JMSTEST@1</ALA_clientId>
      <MTOSI_objectType>ScriptExecutionEvent</MTOSI_objectType>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <ScriptExecutionEvent>
        <scriptName>test</scriptName>
        <scriptType>test</scriptType>
        <scriptVersion>1</scriptVersion>
        <scriptStatus>success</scriptStatus>
        <scriptOutput></scriptOutput>
        <scriptError></scriptError>
      </ScriptExecutionEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

```

    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <scriptExecutionEvent>
        <targetScriptResultFullName>script-manager:script-4:target-script-6:target-script-result-1173467653236</targetScriptResultFullName>
        <fileName>scriptResult/target-script-6_1173467653236.txt</fileName>
        <status>Unknown</status>
        <errorMessage>Trying to connect to an already connected session.</errorMessage>
        <failedCommands />
      </scriptExecutionEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

StateChangeEvent

Table 4-6 describes the different state change events.

Table 4-6 State change events

Event type	Description
AlarmInformationLoaded	Sent to indicate a change in the status of the alarm list; see the XML schema for the possible status values
AlarmCountChanged	Sent when the number of alarms crosses above or below the maximum number of alarms defined in the server configuration
JmsMissedEvents	Sent to inform both durable and non-durable subscribers that events have been missed
SystemInfoEvent	Sent each time a client connects to a new subscription, or reconnects to an existing durable subscription

Code 4-20 shows sample event output for the JmsMissedEvents class.

Code 4-20: Sample JmsMissedEvents output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>JmsMissedEvents</eventName>
      <MTOSI_osTime>1222891102168</MTOSI_osTime>
      <ALA_clientId>oss_client@1</ALA_clientId>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <MTOSI_objectType>StateChangeEvent</MTOSI_objectType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_isVessel>false</ALA_isVessel>
      <ALA_allomorphic/>
      <ALA_eventName>JmsMissedEvents</ALA_eventName>
      <MTOSI_objectName/>
      <ALA_OLC>0</ALA_OLC>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <stateChangeEvent>
        <eventName>JmsMissedEvents</eventName>
        <state>jmsMissedEvents</state>
      </stateChangeEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

```

    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

Code 4-21 shows sample event output for the SystemInfoEvent class.

Code 4-21: Sample SystemInfoEvent output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>SystemInfoEvent</eventName>
      <MTOSI_osTime>1224527204299</MTOSI_osTime>
      <ALA_clientId>oss_client@1</ALA_clientId>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <MTOSI_objectType>StateChangeEvent</MTOSI_objectType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_isVessel>false</ALA_isVessel>
      <ALA_allomorphic/>
      <ALA_eventName>SystemInfoEvent</ALA_eventName>
      <MTOSI_objectName/>
      <ALA_OLC>0</ALA_OLC>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <stateChangeEvent>
        <eventName>SystemInfoEvent</eventName>
        <state>systemInfo</state>
        <sysStandbyIp>192.168.182.253</sysStandbyIp>
        <sysPrimaryIp>192.168.169.94</sysPrimaryIp>
        <jmsStartTime>1224525628189</jmsStartTime>
        <sysStartTime>1224525628189</sysStartTime>
        <sysType>Redundant</sysType>
      </stateChangeEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

StatsEvent

Code 4-22 shows sample event output for the StatsEvent class.

Code 4-22: Sample StatsEvent output

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1170864259082</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>StatsEvent</MTOSI_objectType>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_category>STATISTICS</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <statsEvent>
        <state>end</state>
        <networkElement>10.1.200.71</networkElement>
        <statsType>accounting</statsType>
        <time>1170864259082</time>
      </statsEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>

```

```
</SOAP:Body>
</SOAP:Envelope>
```

TerminateClientSessionEvent

Code 4-23 shows sample event output for the TerminateClientSessionEvent class.



Note – The JMS message terminateClientSessionEvent is broadcast to all sessions.

Code 4-23: Sample TerminateClientSession output

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <MTOSI_osTime>1138395709210</MTOSI_osTime>
      <ALA_clientId />
      <MTOSI_objectType>TerminateClientSession</MTOSI_objectType>
      <MTOSI_NTType>NT_STATE_CHANGE</MTOSI_NTType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_allomorphic />
      <MTOSI_objectName />
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <terminateClientSessionEvent>
        <clientId>ossi@1</clientId>
      </terminateClientSessionEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

XMLFilterChangeEvent

Code 4-24 shows sample event output for the XMLFilterChangeEvent class.

Code 4-24: Sample XMLFilterChangeEvent output

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>XMLFilterChangeEvent</eventName>
      <ALA_category>GENERAL</ALA_category>
      <ALA_OLC>0</ALA_OLC>
      <ALA_isVessel>false</ALA_isVessel>
      <ALA_allomorphic/>
      <MTOSI_osTime>1308255929252</MTOSI_osTime>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <MTOSI_objectName/>
      <ALA_clientId>JMSTEST@1</ALA_clientId>
      <MTOSI_objectType>XMLFilterChangeEvent</MTOSI_objectType>
      <ALA_eventName>XMLFilterChangeEvent</ALA_eventName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
```

```

<jms xmlns="xmlapi_1.0">
  <xmlFilterChangeEvent>
    <filter-Set>
      <filter class="fm.AlarmObject">
        <or>
          <wildcard name="affectedObjectFullName"
value="network:ENB_10.10.13.200:eNBEquip:eNBInst:subscAnd-0%"/>
          <equal name="affectedObjectClassName"
value="netw.NetworkElement"/>
          <equal name="affectedObjectClassName"
value="vpls.L2AccessInterface"/>
          <wildcard name="additionalText"
value="%Down%"/>
          <wildcard name="nodeName" value="%RAN17%"/>
          <in name="nodeId" value="[ENB_10.10.13.200,
ENB_10.10.14.204, ENB_10.10.13.120, ENB_10.10.12.40,
ENB_10.10.14.50, ENB_10.10.14.7]"/>
        </or>
      </filter>
      <filter>
        <or>
          <equal name="networkElement"
value="192.168.3.145" class="jmsEvent"/>
          <equal name="MTOSI_objectType"
value="StatsEvent" class="jmsEvent"/>
        </or>
      </filter>
    </filter-Set>
  </xmlFilterChangeEvent>
</jms>
</SOAP:Body>
</SOAP:Envelope>

```

4.7 JMS message filtering

A JMS subscription must include a filter that, based on JMS header properties, defines the types of messages that the client is to receive. The filter uses SQL syntax. See [“Simple message filters”](#) in this section for more information.

An OSS client that subscribes to the 5620-SAM-topic-xml-filtered topic can use the 5620 SAM-O to define advanced filters based on header and contained-object properties. These optional filters use criteria in XML format. See [“Advanced message filters”](#) in this section for more information.



Note — Alcatel-Lucent recommends that you create filters that are as restrictive as possible so that only the required events are sent. If the filter is not restrictive enough, additional events are queued and processed, which can cause the following:

- increased probability of queue overflows, which result in missed events; see [“Missed events”](#) in section 4.9 for more information
- wasted bandwidth
- degraded 5620 SAM and OSS performance

Mandatory events

You must ensure that several JMS events that an OSS client must process are not inadvertently filtered. Verify that the following mandatory events are not filtered:

- KeepAliveEvent
- StateChangeEvent
- TerminateClientSessionEvent
- JmsMissedEvent
- all events destined for the client ID

The following sample filter ensures that all mandatory events are included.

```
ALA_clientId in ('ossname@1', '')
AND (
    (<filter to include events of specific interest to the OSS>)
    OR
    MTOSI_objectType in ('KeepAliveEvent', 'StateChangeEvent',
'TerminateClientSession')
)
```

Simple message filters

A simple message filter is defined when a subscription is created. To change the filter, a client must unsubscribe and then resubscribe. This type of filter is a string of up to 3000 characters that uses the same syntax as the WHERE clause in an SQL query, as defined in the ANSI SQL-92 standard. The following is an example of a simple filter:

```
ALA_clientId in ('ossname@1', '')
```

This filter includes all of the events in the topic that are intended for the client called ossname@1, and all events in the topic that are intended for all subscribers.



Note — Each subscription must filter on the client ID; otherwise, the subscription is rejected.

A simple filter can include any JMS header properties. Some common properties used for filtering include the following:

- MTOSI_NTType, the notification type
- ALA_category, the event category
- MTOSI_objectType, the object type

See Table 4-3 for a complete list of properties, and Table 4-4 for properties that are specific to alarm events. Each property is formally defined in xmlApiJmsHeader.xsd in the OSS XML schema. See chapter 8 for more information about using the XML schema.

Table 4-7 lists examples of simple JMS message filters.

Table 4-7 Simple JMS filter samples

JMS filter	Description
ALA_clientId in ('ossi@2', ")	To send general messages, and messages qualified with a client ID, directly to the specified client. This filter blocks messages sent to other clients. You must specify this information in all XML topic-based filters.
ALA_clientId in ('ossi@2', ") and ALA_category = 'FAULT' and MTOSI_perceivedSeverity = 'Warning'	To filter fault messages with a warning severity class. The filter excludes some mandatory events. See “Mandatory events” in this section for more information.
ALA_clientId in ('ossi@2', ") and ALA_category not in ('STATISTICS', 'ACCOUNTING')	To stop client statistics, accounting, and keep-alive messages
ALA_clientId in ('ossi@2', ") and (MTOSI_NTType in ('NT_ATTRIBUTE_VALUE_CHANGE', 'NT_OBJECTDELETION', 'NT_OBJECTCREATION') and MTOSI_objectType in ('equipment.PhysicalPort', 'equipment.DaughterCard', 'equipment.BaseCard'))	To include only object creation, object deletion, and attribute changes for cards and ports. The filter excludes some mandatory events. See “Mandatory events” in this section for more information.
ALA_clientId in ('ossi@2', ") and (MTOSI_NTType in ('NT_OBJECTDELETION', 'NT_OBJECTCREATION') and MTOSI_objectType like 'equipment.%')	To include only object creation and deletion for objects in the equipment package. The filter excludes some mandatory events. See “Mandatory events” in this section for more information.

Optional properties

Most JMS header properties apply to all events. However, some alarm-related properties, for example, ALA_alarmType, apply only to specific events. For example, the following filter includes all events except equipment alarms:

```
ALA_clientId in ('ossname@1', '') and ALA_alarmType not in ('equipmentAlarm')
```

The filter also excludes all events that do not include the ALA_alarmType property.

The following filter excludes events with an ALA_alarmType of equipmentAlarm, and includes all other events:

```
ALA_clientId in ('ossname@1', '') and (ALA_alarmType is null or  
ALA_alarmType not in ('equipmentAlarm'))
```

Alarm-specific properties do not apply to all alarm events. When you create a filter using alarm-specific properties, you must include the “is null” case.

The following Boolean example is a filter to exclude correlated alarms, but include alarms that do not have the ALA_isCorr property.

```
ALA_clientId in ('ossname@1', '') and (ALA_isCorr is null or ALA_isCorr  
= true)
```

Advanced message filters

You can create advanced filters to reduce the number of JMS event messages sent to OSS clients. An advanced JMS message filter is a 5620 SAM-O XML filter that specifies the following:

- the types of event notifications that are to be sent to the client
- a set of object properties as criteria for messages sent to the client

Advanced message filters can be used by OSS clients that subscribe to the 5620-SAM-topic-xml-filtered topic. Clients that subscribe to this topic must register a filter using an XML API script that specifies the client ID and filter criteria. To change the filter criteria, the client must register a new filter, which overwrites the previously registered filter.



Note — If an OSS client registers a filter but does not subscribe to the 5620-SAM-topic-xml-filtered topic within 10m, the filter is deregistered.

Registration

To register a filter, clients must use the registerNotification method, which has the following parameters:

- client ID, which is associated with the 5620 SAM-O username (required)
- list of filter criteria, in XML format (required)
- extraTags tag, to specify the JMS header properties to return (optional)



Caution — The registerNotification method entry in the *General Methods/Types* page of the *5620 SAM-O XML Reference* defines the maximum allowed number of leaf elements in a filter. To avoid degraded system performance, Alcatel-Lucent recommends that you consider the current 5620 SAM system load, as well as this limit, before you construct a filter with a large number of leaf elements.

Code 4-25 is an example of a registration message sent by an OSS to the 5620 SAM.

Code 4-25: Registration message

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <security>
        <user>oss_user_1</user>
        <password>MD5-hashed_password</password>
      </security>
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <registerNotification xmlns="xmlapi_1.0">
      <jmsClientId>JMS_CLIENT@4</jmsClientId>
```



```
<filter-Set>
  <filter>
    <between class="service.Service" name="serviceId"
first="150" second="200"/>
  </filter>
  <filter>
    <wildcard class="netw.NetworkElement"
name="outOfBandAddress" value="192.168.101%"/>
  </filter>
  <filter>
    <equal class="jmsEvent" name="ALA_category"
value="FAULT"/>
  </filter>
</filter-Set>
</registerNotification>
</SOAP:Body>
</SOAP:Envelope>
```

The filters in the example are leaf filters that are evaluated using the OR function. The 5620 SAM sends only JMS events that match, or contain an object that matches one or more of the following criteria:

- a service ID between 150 and 200
- an NE with an out-of-band management IP address in the 192.168.101 subnet
- an alarm

To deregister a filter, clients must use the deregisterNotification method, which requires only the OSS client ID. Code 4-26 is an example of a deregistration message.

Code 4-26: Deregistration message

```
<SOAP:Body>
  <deregisterNotification xmlns="xmlapi_1.0">
    <jmsClientId>JMS_CLIENT@4</jmsClientId>
  </deregisterNotification>
</SOAP:Body>
```

If an OSS client unsubscribes from the '5620-SAM-topic-xml-filtered' topic, the 5620 SAM automatically deregisters the client. If a non-durable OSS client connection drops, the client is deregistered after 2m; durable clients are not deregistered.

Event vessels

To further reduce the volume of JMS traffic, most event messages for the 5620-SAM-topic-xml-filtered topic are sent in containers called event vessels. An event vessel is a type of JMS message that contains multiple events but has only one JMS header.



Note 1 – Although the JMS headers of individual events are not sent in an event vessel, you can filter the properties of the headers by specifying “class=jmsEvent” in a filter. See [“Class-based filtering”](#) in this section for more information.

Note 2 – You can configure the 5620 SAM to return JMS header properties of individual events in the event bodies by including the extraTags tag in a registration message. See [“Registration using extraTags”](#) in this section for more information.

The following events are sent on the 5620-SAM-topic-xml-filtered topic but are not sent in an event vessel:

- DBActivityEvent
- DBConnectionStateChangeEvent
- DBProxyStateChangeEvent
- EventVessel
- KeepAliveEvent
- StateChangeEvent
- TerminateClientSessionEvent
- XMLFilterChangeEvent

Table 4-8 lists the event classes that support advanced filtering based on the event properties, the properties of the objects associated with the event, or both:

Table 4-8 Filterable event classes

Event	Event properties	Object properties
AlarmStatusChangeEvent	✓	
AttributeValueChangeEvent	✓	✓
DeployerEvent	✓	
ExceptionEventXMLFormat	✓	✓
FileAvailableEvent	✓	
IncrementalRequestEvent	✓	
LogFileAvailableEvent	✓	
ManagedRouteEvent	✓	
ObjectCreationEvent	✓	✓
ObjectDeletionEvent	✓	✓
RelationshipChangeEvent	✓	
ScriptExecutionEvent	✓	
StatsEvent	✓	

EventVessel messages are sent to a client only after the 5620 SAM acknowledges the successful client registration by sending an XMLFilterChangeEvent message, which contains the registered filter information. Code 4-27 is an example of an EventVessel message that contains the following:

- AttributeValueChangeEvent
- StatsEvent
- ExceptionEvent

Code 4-27: Event vessel message

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>EventVessel</eventName>
      <ALA_category>GENERAL</ALA_category>
      <ALA_vesselSize>4</ALA_vesselSize>
      <ALA_isVessel>true</ALA_isVessel>
      <MTOSI_osTime>1308318280367</MTOSI_osTime>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_clientId>JMSTEST@1</ALA_clientId>
      <ALA_eventName>EventVessel</ALA_eventName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <eventVessel>
        <attributeValueChangeEvent>
          <fullClassName>fm.AlarmObject</fullClassName>
          <objectFullName>faultManager:network@192.168.183.13|a
alarm-31-4-24</objectFullName>
          <attribute>
            <attributeName>numberOfOccurencesSinceClear</attri
buteName>
            <newValue>
              <long>3</long>
            </newValue>
            <oldValue>
              <long>2</long>
            </oldValue>
          </attribute>
        </attributeValueChangeEvent>
        <statsEvent>
          <MTOSI_osTime>1311602579077</MTOSI_osTime>
          <state>begin</state>
          <networkElement>192.168.183.13</networkElement>
          <statsType>accounting</statsType>
          <time>1308256029287</time>
        </statsEvent>
        <statsEvent>
          <MTOSI_osTime>1311602582098</MTOSI_osTime>
          <state>end</state>
          <networkElement>192.168.183.13</networkElement>
          <statsType>accounting</statsType>
          <time>1308256029302</time>
        </statsEvent>
      </eventVessel>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

```

        </statsEvent>
        <ExceptionEventXMLFormat>
            <className>lte.SubscAndEquipmentTraces</className>
            <operation>Deployment to ENB_10.10.12.44</operation>
            <objectName>network:ENB_10.10.12.44:eNBEquip:eNBInst:
subscAnd-0</objectName>
            <requestId/>
            <errorMessage/>
        </ExceptionEventXMLFormat>
    </eventVessel>
</jms>
</SOAP:Body>
</SOAP:Envelope>

```

Filter elements

An advanced filter can contain a combination of leaf and composite filter elements. Leaf elements define an arithmetic or string function, and do not have child elements. Composite filters use Boolean AND, NOT, and OR expressions to evaluate child elements, which are leaf or other composite elements.



Note — Alcatel-Lucent recommends that you create filters that have a top-down hierarchical structure to make them as restrictive as possible. This means that a filter lists the higher-level object properties before the lower-level properties, as defined in the 5620 SAM object hierarchy. See the *5620 SAM-O XML Reference* for object hierarchy information.

Table 4-9 describes the leaf filter types.

Table 4-9 Leaf filter types

Filter element	Description	Example
equal	Return true if the value is the same as the specified value.	<equal name="localAS" value="20"/>
notEqual	Return true if the value is not the same as the specified value.	<notEqual name="localAS" value="20"/>
greater	Return true if the numeric value is greater than the specified value.	<greater name="localAS" value="20"/>
greaterOrEqual	Return true if the numeric value is greater than or equal to the specified value.	<greaterOrEqual name="localAS" value="20"/>
less	Return true if the numeric value is less than the specified value.	<less name="localAS" value="20"/>
lessOrEqual	Return true if the numeric value is less than or equal to the specified value.	<lessOrEqual name="localAS" value="20"/>
anyBit	Return true if an AND operation on the value bit and the specified value does not return 0. The filter is true if any of the "1" bits in the specified value are enabled in the property value. The function applies only to non-negative numeric types and bitmask types.	<anyBit name="parameter" value="3"/>

(1 of 2)

Filter element	Description	Example
allBits	Return true if an AND operation on the value bit and the specified value returns the specified value. The filter is true if all of the “1” bits in the specified value are enabled in the property value. This applies only to non-negative numeric types and bitmask types.	<code><allBits name="parameter" value="5"/></code>
wildcard	Compares values using wildcards. The % character specifies 0 or more characters of any type, and the _ character specifies one character of any type.	<code><wildcard name="SiteId" value="1.%.%.%"/></code>
in	Return true if a numeric or string value equals one of the values in the specified comma-separated list.	<code><in name="parameter" value="[Edge_14, Aggregation_14, Core_14]"/></code>
subset	Return true if a string value is a subset of the specified string.	<code><subset class="service.Site" name="tier" value="123"/></code>
superset	Return true if a string value contains the specified string.	<code><superset class="service.Site" name="siteName" value="Core_"/></code>
notSuperset	Return true if a string value does not contain the specified string.	<code><notSuperset class="service.Site" name="siteName" value="Edge_"/></code>
between	Return true if the numeric value is between the first and second specified values	<code><between name="preference" first="150" second="200" /></code>
past	Return true if a time value is greater than the specified time.	<code><past time="1170864259082" /></code>
empty	Match everything; this is the same as specifying nothing between the <filter> and </filter> tags.	<code><empty /></code>

(2 of 2)

Code 4-28 is an example of an advanced message filter set that contains two filters. Each filter contains leaf and composite filter elements.

The first filter specifies the abstract service.Service class to filter events associated with all service types, and the filter elements specify events that meet the following criteria:

- subscriber ID associated with the service is 22
- AND
- service ID is greater than 10
- AND
 - Administrative State parameter value is 2 or Partially Down
 - OR
 - Aggregated Operational State value is 3 or Down
 - OR
 - Service Tier value is 2

Code 4-28: Advanced filter with leaf and composite elements

```

.
.
.
<filter-Set>

```

```
<filter class="service.Service">
  <and>
    <equal name="subscriberId" value="22"></equal>
    <greater name="serviceId" value="10"></greater>
    <or>
      <equal name="administrativeState" value="2"></equal>
      <equal name="aggrOperationalState"
value="3"></equal>
      <equal name="tier" value="2"></equal>
    </or>
  </and>
</filter>
<filter>
  <and>
    <wildcard name="objectFullName"
value="network:10.20.40.218:shelf-1:cardSlot-1:card:daughterCardSlot
-1:daughterCard:%"></wildcard>
    <not>
      <in class="equipment.Equipment"
name="compositeEquipmentState" value="5,6,7,9"></in>
    </not>
  </and>
</filter>
</filter-Set>
.
.
.
```

The second filter specifies the forwarding of events associated with any port on card 1 of NE 10.20.40.218 that has any compositeEquipmentState except the following:

- 5—equipmentMissing
- 6—equipmentMismatch
- 7—equipmentAdministrativelyDown
- 9—containingEquipmentInTest

Class-based filtering

A class= tag in a filter element specifies the object type to evaluate. You can specify the jmsEvent class to indicate that the filter element is for the event. If a class value is not specified or if the class value is jmsEvent, class-based filtering is not performed. See chapter 12 for information about working with classes.

When you specify `jmsEvent` as the class value, you can filter on the following JMS header properties:

- `ALA_alarmType`
- `ALA_allomorphic`
- `ALA_application`
- `ALA_category`
- `ALA_eventName`
- `ALA_isCorr`
- `ALA_OLC`
- `ALA_routeManager`
- `ALA_span`
- `eventName`
- `fullClassName`
- `MTOSI_aliasNameList`
- `MTOSI_NTType`
- `MTOSI_objectName`
- `MTOSI_objectType`
- `MTOSI_osTime`
- `MTOSI_perceivedSeverity`
- `MTOSI_probableCause`
- `MTOSI_serviceAffecting`



Note – The OSS client user span of control defines the objects that are sent to the client. Using `ALA_span` in a filter is not required unless you want to restrict the returned objects to a subset of the objects within the span of control, or to a specific span such as the Edit span.

When you specify `jmsEvent` as the class, you can also filter on event detail properties. Table 4-10 lists the `jmsEvent` detail properties that are filterable.

Table 4-10 Filterable `jmsEvent` class detail properties

Event detail	Properties
DeployerEvent	successList failedList
ExceptionEvent	objectName operation taskName errorMessage
FileAvailableEvent	fileName
IncrementalRequestEvent	incrementalAction serviceType
LogFileAvailableEvent	serverIpAddress neld
ManagedRouteEvent	errorMessage
RelationshipChangeEvent	changeType fromObjectName fromObjectClass toObjectName toObjectClass

(1 of 2)

Event detail	Properties
ScriptExecutionEvent	targetScriptResultFullName fileName failedCommands systemInvokationId
StatsEvent	networkElement statsType

(2 of 2)

Registration using extraTags

By default, an event vessel does not contain the JMS header properties of the individual events. If you want one or more event-header properties to be returned in an event vessel, you must specify the properties using the `extraTags` tag during registration. The properties are then included in the body of each associated event.

The properties are specified using the following syntax:

```
<tag name="JMS_header_property" eventName="JMS_event_class" />
```

where

JMS_header_property is one of the JMS header properties listed in “[Class-based filtering](#)” in this section

JMS_event_class is one of the event classes listed in [Table 4-8](#)

If a JMS event does not have a specified header property, the property is excluded from the event body. If an `eventName` value is omitted, the specified JMS header property is returned for all events that have the property.

Code 4-29 is an example of a filter registration message that includes an `extraTags` specification.

Code 4-29: Filter registration with extraTags entry

```
<SOAP:Body>
  <registerNotification xmlns="xmlapi_1.0">
    <jmsClientId>JMSTEST@3</jmsClientId>
    <filter-Set>
      <filter>
        <or>
          <equal class="jmsEvent" name="networkElement"
value="192.168.101.145" />
          <equal class="jmsEvent" name="MTOSI_objectType"
value="StatsEvent" />
        </or>
      </filter>
    </filter-Set>
    <extraTags>
      <tag name="MTOSI_osTime" eventName="StatsEvent" />
      <tag name="fullClassName" />
    </extraTags>
  </registerNotification>
</SOAP:Body>
```


In the Code 4-29 example, each StatsEvent message must include the MTOSI_osTime property. All events must include the fullClassName property. However, because a StatsEvent message does not have a fullClassName property, the fullClassName property is not included in the StatsEvent message body, as shown in Code 4-30.

Code 4-30: Event vessel containing extraTags properties

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>EventVessel</eventName>
      <ALA_category>GENERAL</ALA_category>
      <ALA_vesselSize>3</ALA_vesselSize>
      <ALA_isVessel>true</ALA_isVessel>
      <MTOSI_osTime>1308318280367</MTOSI_osTime>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <ALA_clientId>JMSTEST@1</ALA_clientId>
      <ALA_eventName>EventVessel</ALA_eventName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <eventVessel>
        <attributeValueChangeEvent>
          <fullClassName>fm.AlarmObject</fullClassName>
          <objectFullName>faultManager:network@192.168.101.145|
alarm-31-4-24</objectFullName>
          <attribute>
            <attributeName>numberOfOccurrencesSinceClear</attri
buteName>
            <newValue>
              <long>3</long>
            </newValue>
            <oldValue>
              <long>2</long>
            </oldValue>
          </attribute>
          <attribute>
            <attributeName>lastTimeDetected</attributeName>
            <newValue>
              <long>1308318277307</long>
            </newValue>
            <oldValue>
              <long>1308318115170</long>
            </oldValue>
          </attribute>
          <attribute>
            <attributeName>numberOfOccurrences</attributeName>
            <newValue>
              <long>3</long>
            </newValue>
            <oldValue>
              <long>2</long>
            </oldValue>
          </attribute>
        </attributeValueChangeEvent>
      </eventVessel>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

```
</attributeValueChangeEvent>
<statsEvent>
  <MTOSI_osTime>1311602579077</MTOSI_osTime>
  <state>begin</state>
  <networkElement>192.168.183.13</networkElement>
  <statsType>accounting</statsType>
  <time>1308256029287</time>
</statsEvent>
<statsEvent>
  <MTOSI_osTime>1311602582098</MTOSI_osTime>
  <state>end</state>
  <networkElement>192.168.183.13</networkElement>
  <statsType>accounting</statsType>
  <time>1308256029302</time>
</statsEvent>
</eventVessel>
</jms>
</SOAP:Body>
</SOAP:Envelope>
```

Statistics

The 5620 SAM collects JMS statistics that include the following:

- all JMS Subscriber Topic statistics for the 5620-SAM-topic-xml-filtered topic
- JMS Subscriber Session statistics for each active subscription to the 5620-SAM-topic-xml-filtered topic; you can use the statistics to monitor the count and size of the filtered event vessels
- Publisher XML Event statistics that are related to advanced JMS message filters; you can use the statistics to monitor the number of filter change events, and the count and size of the published event vessels

See the *5620 SAM Statistics Management Guide* for more information about the statistics counters that are related to JMS message filtering.

4.8 JMS and redundancy

JMS subscribers in a redundant system must be notified of switchover events and disconnections of clients. Each time a JMS subscriber connects to a topic, a message is sent containing the current system information. The ID of the connected client is included in the ALA_clientId header property. OSS developers must verify the client ID to ensure that the message is specific to the client, as this message is sent to all clients on that topic. This event is a StateChangeEvent with a state of 8, which represents System Information. Table 4-11 lists the properties in a System Information StateChangeEvent message.

Table 4-11 StateChangeEvent System Information properties

Property name	Description
jmsStartTime	UNIX time of most recent JMS-server restart; long integer
sysPrimaryIp	IP address of primary main server; string

(1 of 2)

Property name	Description
sysStandbyIp	IP address of standby main server in redundant 5620 SAM deployment; string
sysStartTime	UNIX time of most recent main-server restart; long integer
sysType	5620 SAM system type; string, value is redundant or standalone

(2 of 2)

Storing System Information properties for the subscriber can help identify events such as the following:

- a server switchover since the previous client connection, represented by a change in the sysPrimaryIp value
- a main-server or JMS-server restart, represented by a change in the sysStartTime or jmsStartTime value

See “[StateChangeEvent](#)” in section 4.6 for more information.

4.9 Connection monitoring and error recovery

There are two scenarios for which an OSS must monitor and, if necessary, recover from:

- loss of connectivity
- loss of events

These scenarios can occur separately or together. For example, when using a durable subscription, a loss of connectivity may occur without a loss of events. Loss of events can also occur when the message system is under heavy load, without connectivity loss, and is flagged by a jmsMissedEvents message. Finally, any loss of connection which also involves a subscription being removed—which can occur in several situations, including but not limited to an activity switch or a server restart.

The manner in which an OSS manages these scenarios depends their specific needs and the expectations of their customers. For example, for some applications, a loss of connectivity should always result in the OSS application reconnecting to the 5620 SAM. In other situations, the desired behavior may depend on the cause of the connectivity loss—for example, if a 5620 SAM administrator intentionally disconnects an OSS, it may not be desirable to immediately reconnect.

How to recover from lost events also depends on the OSS. If an OSS uses JMS events to maintain a local information store that mirrors some data set in the 5620 SAM—such as a network inventory or a list of current alarms—then a loss of events results in the OSS being out of sync with the 5620 SAM.

The following scenarios could be used to determine whether the OSS needs to resync the database of inventory and alarm information:

- If a durable OSS disconnects but remains subscribed to the 5620 SAM, on reconnect if a `JmsMissedEvent` message is not received and the `sysStartTime` is not changed, it is not necessary to do a resync of inventory and alarm information because all events are queued on the 5620 SAM server and await OSS reconnection.
- If a disconnected durable subscription is removed from an active 5620 SAM server, a `JmsMissedEvents` message will be generated when the client reconnects using the same client ID. This indicates to the OSS that a resync of inventory and alarm information needs to be performed because events have been missed.

After the OSS detects it is out of sync, possible recovery scenarios could include:

- immediately resynchronizing with the 5620 SAM, such as by retrieving inventory information or the latest alarm list through the XML API
- notifying the OSS administrator of the problem, and allowing the administrator to select a recovery approach—for example, an immediate resync or a scheduled resync for a later time

When you implement a recovery procedure, you must consider that events will continue to occur in the network while an OSS is busy resynchronizing or populating its database for the first time. For this reason, an OSS must process events while they are resynchronizing, and may need to recover from error conditions that require them to reconnect or restart the resync.

Procedure 4-1 To resync an OSS database with the 5620 SAM

- 1 Establish a JMS subscription, listen for events, and buffer events locally. Alcatel-Lucent recommends that the OSS maintain two local buffer queues:
 - one queue for critical system-related events—such as the `KeepAliveEvent`, `SystemInfoEvent`, or `JmsMissedEvent`—which the OSS can use to monitor the state of the JMS connection.
 - one queue for inventory or alarm events in which the OSS is interested.



Note — The OSS must not begin processing the events from the second buffer until step 3 is successfully completed.

- 2 Begin retrieving inventory information or the latest alarm list.



Note — While inventory is being retrieved, the OSS must monitor the first buffer queue, as described in step 1. If the OSS detects connectivity loss with the 5620 SAM, or a loss of JMS events, the OSS should terminate the current inventory retrieval process, if possible. The internal buffer queues should be cleared. Perform recovery procedures to restore JMS connectivity, if necessary, then repeat step 2.

- 3 After the inventory retrieval completes successfully, populate the OSS database with the buffered events from step 1.
-

JMS exception

You must use JMS exceptions if you have a JMS connection to a 5620 SAM server. JMS internally monitors client connections to the JMS server and throws an exception if a connection between the JMS server and a client is lost. You must implement the `javax.jms.ExceptionListener` interface on the JMS client to enable the monitoring of exceptions. The interface contains a call-back method that is invoked for all exceptions. A typical implementation of the call-back method attempts to reconnect and possibly take another action, such as generating an event.

Monitoring for incoming events

In addition to handling exceptions, an OSS must monitor incoming events to ensure that the JMS connection is active. A `KeepAliveEvent` is published at approximately 30-second intervals to each of the JMS topics even when no other messages are sent.

`KeepAliveEvents` may be received ahead of other event types.

If no events are received within a reasonable time period, you must investigate the status of the 5620 SAM server.

Events may be subject to latency issues related to network throughput, the volume of events being processed on a 5620 SAM main server, and the processing speed of the client. If the client does not receive any expected events within a reasonable period, such as 5m, you must investigate the cause of the excessive latency. Potential problems can be related to the 5620 SAM main server status. See the *5620 SAM Troubleshooting Guide* for information about how to check the server status, or contact your Alcatel-Lucent OIPS technical-support representative for information about how to diagnose the problem and implement corrective actions.



Note 1 — You must define the JMS filter to include keep-alive messages. See section 4.7 for more information.

Note 2 — To prevent the loss of keep alive messages, you must ensure that the system clocks of the OSS clients and 5620 SAM main server are synchronized.

5620 SAM-O session termination

You can use the 5620 SAM GUI client to close and remove a durable subscription when you no longer require a durable client. See the *5620 SAM User Guide* for more information about removing durable subscriptions.

The 5620 SAM-O uses the following JMS event to notify OSS client applications of a session termination:

`TerminateClientSession`

The `TerminateClientSession` event indicates that a client JMS session is about to be closed. The client must clean up the disconnected session when the message is received. Additional session termination behavior is dependent on the requirements of your OSS client application. For example, you can also configure the requirement to close the OSS client application.

Missed events

Both durable and non-durable subscriptions can be used by OSS clients that cannot tolerate losing events without taking corrective action. However, in certain situations, events can be missed and subscribers are notified with a `JmsMissedEvents` message.

JMS messages are queued in the 5620 SAM until they are acknowledged by the JMS consumer, or 5620 SAM-O client. If the JMS message queue overflows, the following occurs for both durable and non-durable JMS subscriptions:

- Connected non-durable subscribers
 - the 5620 SAM stops queuing messages until there is capacity in the queues
 - a `JmsMissedEvents` message may or may not be sent depending on the internal cause of the overflow and whether there is capacity in the queues.
- Connected durable subscribers
 - queued messages are removed
 - a `JmsMissedEvents` message is added to the queue. See code 4-31 for more information.
 - new events continue to be queued or sent
- Non-connected non-durable subscribers
 - Not applicable since a non-connected non-durable subscriber becomes unsubscribed as soon as it disconnects from the 5620 SAM server.
- Non-connected durable subscribers
 - queued messages are removed
 - the subscriber is unsubscribed
 - no new events are queued

The following are examples for which messages may be missed:

- the client is slow, the message rate is high, or there is high latency in the network
- disconnected clients with active subscriptions, resulting in messages being queued until the client reconnects

Table 4-12 describes the alarms that are related to JMS clients and messages.

Table 4-12 JMS-specific alarms

Alarm	Description
JMSDurableClientReset	Raised when a durable JMS client is reset as the result of a JMS server restart or activity switch
JMSClientMessagesRemoved	Raised when a JMS client has messages removed after exceeding the configured message limit. This applies to OSS durable subscribers only.
JMSDurableClientUnsubscribed	Raised when a durable JMS client is automatically unsubscribed. This occurs when a disconnected durable client exceeds the configured message limit.

JmsMissedEvents message

The 5620 SAM sends a JmsMissedEvents message to indicate that events have been lost, allowing subscribers to detect missed events. The JmsMissedEvents message is a StateChange Event, as shown in Code 4-31.

Code 4-31: JmsMissedEvents message

```
<SOAP:Envelope
xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>JmsMissedEvents</eventName>
      <MTOSI_osTime>1222891102168</MTOSI_osTime>
      <ALA_clientId>oss_client@1</ALA_clientId>
      <MTOSI_NTType>ALA_OTHER</MTOSI_NTType>
      <MTOSI_objectType>StateChangeEvent</MTOSI_objectType>
      <ALA_category>GENERAL</ALA_category>
      <ALA_isVessel>false</ALA_isVessel>
      <ALA_allomorphic/>
      <ALA_eventName>JmsMissedEvents</ALA_eventName>
      <MTOSI_objectName/>
      <ALA_OLC>0</ALA_OLC>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <stateChangeEvent>
        <eventName>JmsMissedEvents</eventName>
        <state>jmsMissedEvents</state>
      </stateChangeEvent>
    </jms>
  </SOAP:Body>
</SOAP:Envelope>
```

Recovery from missed events

Clients must be able to recognize when events are missed and take recovery action that is appropriate for their application. For example, in an OSS application for which events are being used to maintain inventory information, the recovery action could include:

- an immediate resync of inventory information
- notification being sent to an administrator, allowing them to perform an immediate resync or schedule a future resync

Although an OSS application must have measures in place to recover from missed events, the OSS application can implement prevention methods. The following are examples of ways an OSS application can prevent missed events:

- use restrictive filtering wherever possible. See section 4.7 for more information.
- increase OSS efficiency in processing events
- minimize the amount of time that an OSS is disconnected from the 5620 SAM, but still subscribed (where events are being queued but not processed)
- queue messages internally to facilitate the handling of temporary message bursts
- use DUPS_OK_ACKNOWLEDGE acknowledgement mode to facilitate the handling of network latency. See “[Acknowledgement modes](#)” in section 4.5 for more information.

4.10 Managing and monitoring JMS sessions

The following subsections describe how to manage and monitor JMS connections.

Monitoring sessions

You can manage and monitor the state of JMS connections and error conditions using the 5620 SAM GUI.

You can view the status of current sessions and, if necessary, close or remove sessions. When you close a non-durable subscription, the client is disconnected—after sending a `TerminateClientSession` event—and the subscription is removed. When you close a durable subscription, a `TerminateClientSession` event is sent and the client is disconnected. However, the subscription is not removed. The session continues to be visible in the session list and the subscription remains active so events are being queued. However, the status indicates that the session is disconnected. You can remove a disconnected session. See procedure 4-2 for more information.

You can also use the `stopOssSession` method in the `security.SamSessionManager` class over HTTP to close a JMS session. You must be an admin user or the user that initiated the session to close the JMS session using this method.

The 5620 SAM raises alarms for events that impact JMS sessions. See Table 4-12 for more information about the alarms that are related to JMS clients and messages.

JMS logging

The 5620 SAM JMS server logs all connections to JMS, including the JMS filter information. The 5620 SAM JMS server records the connection information in the `jmsServer.log` file. To obtain the log file, go to the 5620 SAM JMS server installation directory and navigate to the `/nms/log/jmsserver` directory. You can then open the log file using the 5620 SAM LogViewer utility. The log file is backed up when it reaches a configured value. You can view the backup files in the same directory as the original file. See the *5620 SAM Troubleshooting Guide* for information about using the 5620 SAM LogViewer utility.

Procedure 4-2 To manage a 5620 SAM-O JMS client session or remove a durable subscription

- 1 Using an account with an assigned security scope of command role, choose Administration→Security→5620 SAM User Security from the 5620 SAM main menu. The 5620 SAM User Security - Security Management (Edit) form opens.
- 2 Click on the Sessions tab button.
- 3 Specify a filter to create a filtered list 5620 SAM-O JMS client sessions and click on the Search button. The list of currently active client sessions opens.
- 4 Choose a JMS session from the list and click on the Close Sessions button to shut down the client session. The connection to the server is terminated when you close a durable JMS session, however, the subscription continues to store JMS messages. When you close a non-durable session, the session is also removed.
- 5 Perform one of the following steps.
 - a To remove a durable subscription and shut down the client session, choose a JMS session from the list and click on the Remove Session button. The server stops storing the JMS messages for the session.



Note — When you remove a durable subscription, the OSS client can still attempt to connect to the 5620 SAM-O server. You can prevent an OSS client from attempting to connect by suspending the OSS user account. See the *5620 SAM User Guide* for more information.

- b Go to step 6.
 - 6 Validate the action.
 - 7 Close the form.
-

4.11 JMS consumer creation

You can use Procedure 4-3 to create a simple JMS consumer. The JMS example code is available from the *5620 SAM-O JMS Example* link on the 5620 SAM on-product documentation home page.

Procedure 4-3 To compile and create a sample JMS client *.java file

To perform this procedure, you require the samOss.jar file for your classpath, which you can obtain from the following locations:

- *path*/integration/SAM_O directory on a 5620 SAM main server where *path* is the main server installation location, typically /opt/5620sam/server
- http://main_server_address:8080/integration/samOss.jar where *main_server_address* is the main server IP address or hostname



Warning — Each OSS client must use the samOss.jar from the current 5620 SAM release. Using the incorrect samOss.jar can cause the 5620 SAM main server or clients to become unstable. The 5620 SAM release information is in the JAR manifest file.



Note — The samOss.jar file contains all of the classes required to connect to the 5620 SAM. If an OSS client must connect to multiple releases of the 5620 SAM, the OSS must run a separate JVM with the samOss.jar that corresponds to the 5620 SAM release. Alternatively, the OSS client can use multiple class loader instances in the same JVM to access multiple versions of the samOss.jar.

The following jars are deprecated; you must use only samOss.jar to connect to the 5620 SAM-O:

- samOssJBoss.jar
Does not contain any application server classes, only specific 5620 SAM classes (for use when running within a JBoss application server).
- samOssAnyServer.jar
Contains the same classes as the samOss.jar, in addition to necessary classes for communication with the JMS server (for use when running within another application server).

It is only necessary to include one jar specific to the OSS application needs. If you choose a jar other than the samOss.jar you must update your classpath accordingly.

- 1 Compile the *.java file. The following code is for a *.java file named JmsTest.java:

```
javac  
  
-classpath .:samOss.jar JmsTest.java
```

You must know the path to the samOss.jar file on the server.



Note — A JMS client must use the same Java version, or later, as the 5620 SAM. See the *5620 SAM Planning Guide* to determine the supported version.

- 2 Run the compiled program:

```
java -classpath .:samOss.jar JmsTest -t <topic> -s <App server IP:port>

{-r <HA App server IP:port>} -u <user> -p <password> -f "<filter>"

{-persistent} -c <uniqueid>
```

where

App Server IP is the IP address of the 5620 SAM server

port is the JMS port number configured during installation

uniqueid, *user*, and *password* are valid account information configured by the 5620 SAM

filter is a valid JMS filter. See section 4.7 for more information.



Note 1 – If the 5620 SAM server is configured to use SSL, you must run JmsTest with the following command:

```
-Djavax.net.ssl.trustStore=<keystore_file>
```

where *keystore_file* is the fully qualified path to the 5620 SAM main server keystore file.

For example:

```
java -classpath.:samOss.jar
-Djavax.net.ssl.trustStore=<keystore_file> JmsTest
```

See the *5620 SAM User Guide* for more information about how to configure SSL.

Note 2 – To ensure that the 5620 SAM server reconnects after an activity switch for HA (High Availability) JMS connections, you must set the HA port for JMS connections to the JBoss High Availability JNDI service port value of 1100.

See the *5620 SAM Planning Guide* for more information about the list of ports required on the 5620 SAM server deployment.

4.12 JMS record and playback tool

You can use the JMS record and playback tool to record JMS messages coming from a 5620 SAM network by capturing and writing the JMS message flow to an XML file. The playback function then takes the messages in the XML file and plays them back to an OSS or the 5620 SAM.



Note 1 – The playback function is a JMS exercise and there is no interaction with the underlying 5620 SAM network or 5620 SAM database.

Note 2 – There is no memory management built into this tool. You must stop the recording of messages to avoid disk partition overflow.

Note 3 – You can modify the messages and delays by adding or moving attributes.

Note 4 – The delay is measured in ms.

Note 5 – Only the syntax is validated. Invalid syntax prevents playback.

Procedure 4-4 To record, stop and playback JMS messages



Warning – The playback function may affect the 5620 SAM main server, GUI client, or OSS client performance.

- 1 Log in to a 5620 SAM main server as the samadmin user.
- 2 Open a console window.
- 3 Enter the following at the prompt:

```
bash$ cd nms/bin/unsupported ↵
```

- 4 Enter the following at the prompt to start recording JMS messages:

```
bash$ ./jmsTools.bash jmsRecorder start filename ↵
```

where *filename* is the absolute path of a file that is to contain the recorded JMS messages

- 5 To stop recording JMS messages, enter the following at the prompt:

```
bash$ ./jmsTools.bash jmsRecorder stop ↵
```

- 6 To play back the recorded JMS messages, enter the following at the prompt:

```
bash$ ./jmsTools.bash jmsReplayer filename delay ↵
```

where

filename is the absolute path of the file that contains the recorded JMS messages

delay is Yes or No, to specify whether to use the delays captured during the recording

5 — XML requests

- 5.1 XML communication with the 5620 SAM server 5-2
- 5.2 Monitoring the status of the 5620 SAM-O connection using XML API ping 5-5
- 5.3 Workflow to set up and operate an HTTP XML request-response connection 5-6
- 5.4 Viewing XML requests ignored by the server 5-7
- 5.5 5620 SAM-O server-specific commands 5-8

5.1 XML communication with the 5620 SAM server

The XML API is accessible using a servlet that runs on the 5620 SAM server. The 5620 SAM-O uses HTTP and HTTPS protocols to transport the XML requests between the client application and the server. See the *5620 SAM User Guide* for more information about how to configure HTTP or HTTPS access.



Warning — The maximum number of concurrent OSS connections is 10 or more and varies based on server hardware. See the *5620 SAM Planning Guide* for information about platform sizing.

In a 5620 SAM network with multiple OSS applications, Alcatel-Lucent recommends that each OSS application use only one HTTP connection for each XML/SOAP request. This helps to avoid having one OSS application consuming all connections.

If more than one HTTP connection is required, for example, for performance and scalability, you must make the number of HTTP connections configurable at run time (after installation), so that system integrators can manage the HTTP connections across the OSS applications that connect to the same 5620 SAM server. Contact your Alcatel-Lucent OIPS technical support representative for more information.



Note — You must use the HTTP POST method to send XML API requests.

PostXML

For the purpose of demonstrating the functionality for sending XML requests to the 5620 SAM server, this subsection uses the PostXML.java application as a simple example. The source code contains methods that post XML requests to a remote web server using the HTTP POST method.



Note — PostXml.java software was developed by voluntary contributions made by individuals on behalf of the Apache Software Foundation as part of the Apache Jakarta Project. Alcatel-Lucent does not support or endorse the postXML.java application, or any other third-party application. For a more robust solution, Alcatel-Lucent recommends developing code that is customized for the OSS application to generate XML and send it over an HTTP connection without creating files for each request. The HTTP connections should be managed directly by the OSS process.

The PostXML.java file includes the actual function that posts the XML code to the 5620 SAM server, and requires the following Java library files:

- commons-httpclient.jar
- commons-logging.jar
- commons-codec.jar

PostXML requires the same version of the files as the server. Table 5-1 lists the locations for Java library files on the 5620 SAM server.

Table 5-1 Location for Java library files

5620 SAM component	Location
server	C:\<5620sam_home>\server\nms\jboss\lib\ or /opt/<5620sam_home>/server/nms/jboss/lib

PostXML is run from within a Java session that is configured with the appropriate class path setting to the 5620 SAM server. Code 5-1 shows a sample batch file configuration that could be used to initialize the Java session.

Code 5-1: PostXML batch file

```
@echo off
set CP=.;commons-httpclient.jar;jecho "classpath=%CP%"
java -classpath %CP% PostXML
http://<ip_address_sam_server>:8080/xmlapi/invoke %1
```

You can reference the PostXML batch file in your request to the 5620 SAM server. Code 5-2 shows a sample command to post a request to create a service.

Code 5-2: Example of how to post a request

```
call PostXML.bat createEpipe.xml
```

The 5620 SAM server responds to the request and uses the PrettyPrint.xsl stylesheet to format the display sent to the client command line interface. The response can contain the following information:

- notification of the successful or failed request
- basic information such as the fully distinguished name for the configured object
- all known information for the configured object

Procedure 5-1 To post an XML request to the 5620 SAM server

You must be a valid user on the 5620 SAM. User and password information is sent with each request to the 5620 SAM server.



Note 1 — You must use a user account with OSS Mgmt permission to access the 5620 SAM-O. See the *5620 SAM User Guide* for more information about the 5620 SAM user accounts and privileges.

Note 2 — This procedure uses the following component as an example.

- PostXML to post the XML requests to the 5620 SAM server. The utility is available from the website for the Apache Jakarta Project.

You can choose a different utility and stylesheet for your 5620 SAM-O implementation.

1 Specify the HTTP or HTTPS access address.

a For HTTP access, use:

```
http://server_name:port#/xmlapi/invoke
```

where

server_name is the host name or IP address of the 5620 SAM server

port# is a port on the 5620 SAM server configured for HTTP or HTTPS, as described in the *5620 SAM | 5650 CPAM Installation and Upgrade Guide*

b For HTTPS access, use:

```
https://server_name:port#/xmlapi/invoke
```

where

server_name is the host name or IP address of the 5620 SAM server

port# is a port on the 5620 SAM server configured for HTTP or HTTPS, as described in the *5620 SAM | 5650 CPAM Installation and Upgrade Guide*

The XML application servlet is running and available from the appropriate port. The SOAP-enabled XML can now be sent to the server. For more information about communication between the OSS and the server, contact your Alcatel-Lucent account or OIPS technical support representative.

2 Put the following files in the C:\<5620sam_home>\client\nms\ directory on the client station:

- PostXML.class
- PostXML.bat
- commons-httpclient.jar
- commons-logging.jar
- commons-codec.jar

3 Create an XML request script that you want to post to the 5620 SAM server.

4 Put the XML request script in the same directory as the 5620 SAM-O client files. The 5620 SAM-O client files are typically in the following directory:

- on a Windows station, C:\5620sam\client\sam-o_client
- on a Solaris station, /opt/5620sam/client/sam-o_client

- 5 Create a script to post the request to the 5620 SAM main server. See Code 5-1 for an example script file.
- 6 Open a CLI session.
- 7 Post the request to the 5620 SAM server by entering the following command:

```
call PostXML.bat <xml_request_name>.xml
```

where `<xml_request_name>` is the name of your XML request

The 5620 SAM server responds with the request execution status and the requested information. See Table 9-2 for more information about the execution status that is provided by the HTTP response codes.

5.2 Monitoring the status of the 5620 SAM-O connection using XML API ping

You can use XML API ping to monitor the HTTP connection to the 5620 SAM web server.



Note — See chapter 4 for more information about using the following methods to monitor the status of the 5620 SAM-O connection:

- JMS exception to monitor client connections to the JMS server
- monitoring near-real-time events using JMS to monitor client connections to the JMS server

XML API ping

The XML API ping method is used to monitor the HTTP connection to the 5620 SAM web server.

Code 5-3 shows an XML API ping that you can use to test the ability of the OSS application to access the 5620 SAM web server. Information in the SOAP/XML request includes the:

- standard SOAP user and password encoding
- ping command to look for the release of XML on the server, in this case version 1.0



Caution — The following sample message is an example of the request format. Use the sample as a base to build your request. Ensure that you test your request before network deployment.

Code 5-3: XML interface check

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope>
  <SOAP:Header>
```

```
<header xmlns="xmlapi_1.0">
  <security>
    <user>user name</user>
    <password>MD5-hashed user password</password>
  </security>
  <requestID>clientName@requestId</requestID>
</header>
</SOAP:Header>
<SOAP:Body>
  <ping xmlns="xmlapi_1.0"/>
</SOAP:Body>
</SOAP:Envelope>
```

Code 5-4 is an example of a successful response message to the ping request. The response indicates that the correct version 1.0 of XML on the server responded to the ping.

Code 5-4: XML interface check response

```
<SOAP:Envelope>
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <pingResponse xmlns="xmlapi_1.0"/>
  </SOAP:Body>
</SOAP:Envelope>
```

An exception response to a failed ping could result in numerous types of return messages. For example:

- socket timeouts
- HTTP 404 errors
- connection exceptions

5.3 Workflow to set up and operate an HTTP XML request-response connection

The following workflow lists the high-level steps required to set up and operate an HTTP XML request-response connection. This workflow assumes that you have completed Procedure 3-2 to set up and operate the 5620 SAM-O.

- 1 Develop the application that requests data from, or sends data to, the server and/or the database. See Procedure 5-1.
 - i Construct the request by specifying:
 - methods that define the executed function
 - package classes used that define the kinds of objects, and the values of the objects (parameters) requested
 - filters to limit or customize the scope of the request
 - information structure and data types
 - ii Send the request. See Procedure 5-1.

- 2 Manage the responses from the server.
 - a View the error or success messages.
 - b Handle the retrieved information appropriately.
- 3 As required, enable logging of ignored XML requests. See Procedure 5-2 for more information.
- 4 Close the session.

5.4 Viewing XML requests ignored by the server

If an OSS application includes an element that is not in the schema in the XML request, the element is ignored and the OSSI processes the request. All XML requests with matching start and end tags are accepted by the OSSI, including spelling and case errors. The 5620 SAM-O logs a message to the serverLogFile.



Note — Enabling all logging for the XML API layer is only suitable for development environments.

Procedure 5-2 To enable server log file logging of ignored XML requests



Caution — The server log file should only be enabled on servers in a development environment. When you enable the server log on a server that is running in live network environment, server performance may be affected.

- 1 Open the nms-server.xml file on the 5620 SAM server. The file is located in the *install directory/nms/config* directory, where *install directory* is the installation directory.
- 2 Find the <Filter> element within the <SystemLog> element.
- 3 Add the following code directly after the line beginning with <!-- Don't change this section END -->

```
<include severity="debug" package=".*xmlapi.*"/>
```



Caution — Do not modify other nms-server.xml parameters. Modifying the file can seriously affect the network management and performance of the 5620 SAM.

- 4 Restart the 5620 SAM server. All XML requests not in the XML schema are now logged to the server log file.
-

5.5 5620 SAM-O server-specific commands

The 5620 SAM-O provides XML commands that allow you to specify completion of the following actions:

- obtain the local server time for the request
- identify the server software load

Obtaining the local server time for a request

You can include the `<timeStamp xmlns="xmlapi_1.0"/>` element in your request to obtain the server time for the request in milliseconds. Code 5-5 shows a sample of the `<timeStamp xmlns="xmlapi_1.0"/>` element and server-specific version commands used in a request. Code 5-6 shows a sample of the response to the request in Code 5-5.



Note — The server time for the request in the SOAP response header indicates when a request stream was opened. It does not indicate when a request completed.

You can insert multiple `<timeStamp xmlns="xmlapi_1.0"/>` elements for a SOAP message that contains multiple message requests. You can use the delta between the timestamp commands to determine the milliseconds required to execute the intermediate commands. See “Requests” in section 9.1 for more information about multiple requests.

Identifying the server software load

You can include the `<version xmlns="xmlapi_1.0"/>` element within the body of your XML request to determine the software load installed on the 5620 SAM server. The server returns the software load information in an XML response with the following format.

```
5620 SAM Version <version>.<major.minor>.<build_number>.<patch_number>
```

The 5620 SAM server does not return patch information in the XML response if there are no issued patches for the server software.

Code 5-5: Sample timestamp and server-specific version request

```
<SOAP:Envelope>
<SOAP:Header>
<header xmlns="xmlapi_1.0">
  <requestID>clientName@requestId</requestID>
</header>
</SOAP:Header>
<SOAP:Body>
  <timeStamp xmlns="xmlapi_1.0"/>
  <ping xmlns="xmlapi_1.0"/>
  <timeStamp xmlns="xmlapi_1.0"/>
  <version xmlns="xmlapi_1.0"/>
  <timeStamp xmlns="xmlapi_1.0"/>
</SOAP:Body>
</SOAP:Envelope>
```

Code 5-6: Sample response to timestamp and server-specific version request

```
<SOAP:Body>
<timestampResponse xmlns="xmlapi_1.0">
  <millis>1168877827625</millis>
</timestampResponse>
  <pingResponse xmlns="xmlapi_1.0" />
<timestampResponse xmlns="xmlapi_1.0">
  <millis>1168877827626</millis>
</timestampResponse>
<versionResponse xmlns="xmlapi_1.0">
<version>5620 SAM Version 0.0.0.0</version>
<baseVersion>0.0</baseVersion>
<build>0</build>
<patch>0</patch>
</versionResponse>
<timestampResponse xmlns="xmlapi_1.0">
  <millis>1168877827627</millis>
</timestampResponse>
</SOAP:Body>
```


5620 SAM-O information model

- 6 – 5620 SAM-O information model overview**
- 7 – 5620 SAM-O XML Reference**
- 8 – 5620 SAM-O XML Schema**
- 9 – XML message structure**

6 — 5620 SAM-O information model overview

6.1 5620 SAM-O information model overview 6-2

6.2 5620 SAM-O XML packages 6-3

6.1 5620 SAM-O information model overview

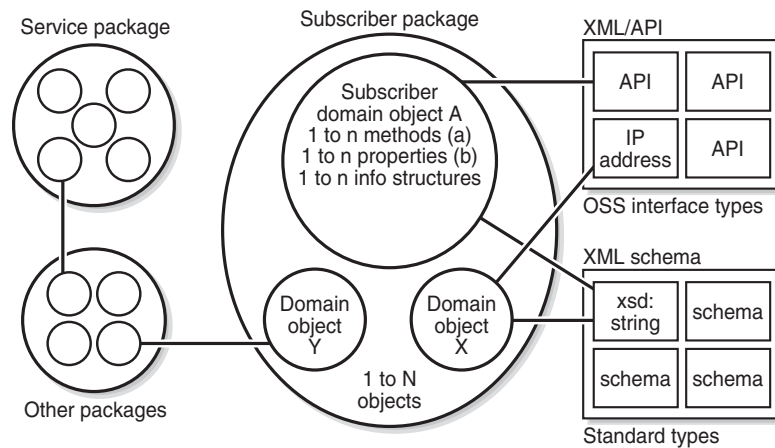
The 5620 SAM-O information model maps to the 5620 SAM java code base. It is defined by an XML schema, which is organized in packages.

The information model consists of:

- packages with one or more groups of related domain objects (classes), data types, bitmasks, and enumerations
- classes that contain:
 - properties of the class
 - information structures for both the domain objects (classes) and properties of the information, which are called elements in the 5620 SAM-O but are known as parameters in CLI and the 5620 SAM GUI
 - NE-specific properties
- relationships between domain objects that indicate the inheritance and containment of these objects
- standard and specific XML schema definitions of types and methods, as well as XML schema definitions specific to 5620 SAM functions.

The information model provides a type of object model of the 5620 SAM-managed network. Figure 6-1 shows an example of the information model in terms of subscriber configuration.

Figure 6-1 Information model - subscriber package



a) specify method

b) specify properties <inparam... >

Notes: Packages organize related objects under the same name.

Lines represent the relationship between objects.

Circles represent an object or package, the squares represent the types.

17303

The material at the following links describes the 5620 SAM-O information model in detail; the links are in the on-product documentation home page:

- *5620 SAM-O XML Reference*—Provides information about the XML interface, including package descriptions, object property values and descriptions, UML inheritance drawings, domain objects, or classes, parent relationships, inherited properties and methods, and supported NE types
- *5620 SAM-O XML Schema*—Provides a definition of the XML schema for each 5620 SAM-O package, and includes a list of the methods and types that are associated with the package

6.2 5620 SAM-O XML packages

Each package provides functionality that OSS applications can access. They can be categorized into functional areas of the 5620 SAM. A few package examples are provided for each area:

- Equipment — fr, netw, equipment, ethernetequipment
- Policies — file, qos, acl, slope, accounting
- Routing Protocols — bgp, ospf, rip, isis
- Services — service, subscr, ies, vpls, vll, vprn
- Generic — generic, user
- Statistics — statistics, log
- Diagnostics — sas, ethernet am
- Faults — fm
- Database — db
- CPAM — topology, rca, monpath

Each package contains methods and types for each domain.

See Procedure [7-1](#) for more information about how to view package information in the *5620 SAM-O XML Reference*.

See Procedure [8-1](#) for more information about how to view package information in the *5620 SAM-O XML Schema*.

7 — 5620 SAM-O XML Reference

7.1 Overview	7-2
7.2 Main menu	7-3
7.3 Package list	7-4
7.4 Classes and types list	7-4
7.5 Class details	7-5
7.6 Type details	7-13
7.7 XML schema changes	7-13
7.8 JMS changes	7-14
7.9 Procedures	7-14

7.1 Overview

The *5620 SAM-O XML Reference* provides information about the XML API specification for the 5620 SAM-O interface. The *5620 SAM-O XML Reference* provides information about all of the packages that define the functionality on various 5620 SAM domains that an OSS application can access. The *5620 SAM-O XML Reference* includes the following:

- package descriptions
- object/class property values and descriptions
- UML inheritance drawings
- general methods and types
- deprecated packages, properties, methods, classes, and enumerations
- lists of supported NEs and NE releases
- NE-specific information about supported properties per NE release

Package, class, type, UML inheritance diagrams, property descriptions, and value details are presented on HTML pages and include the following:

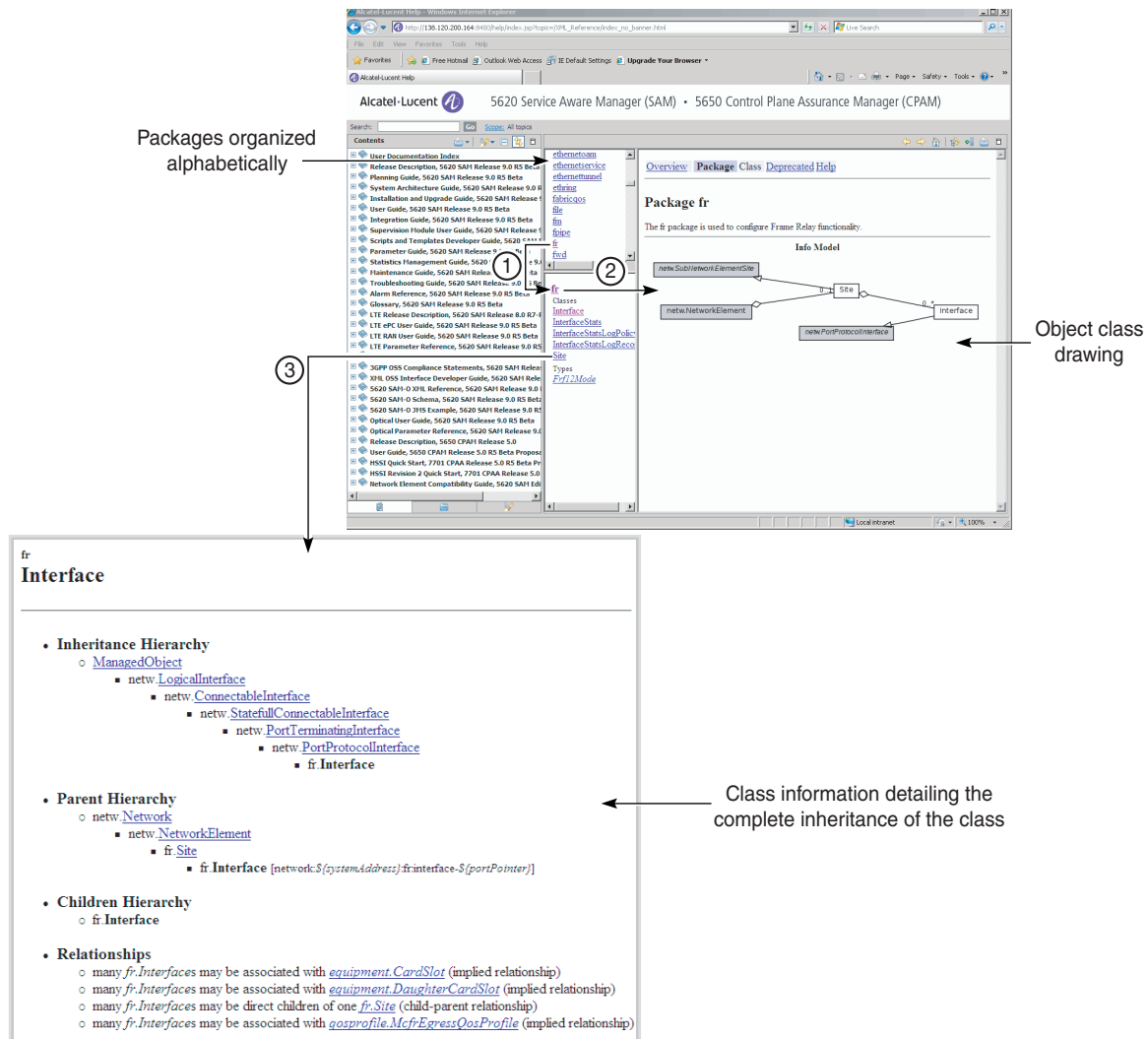
- package list—displays the packages that are available to the XML interface (upper left panel)
- class list—displays the classes and types that are defined for each package (lower left panel)
- class details—contains information about each class inheritance, relationships, class type, properties, methods, and supported NE types and releases (right panel)

Deprecated properties, general methods/types, and supported products are all available from the upper left panel of the *5620 SAM-O XML Reference*. See Procedures [7-4](#), [7-5](#), and [7-6](#) for more information.

Perform Procedure [7-1](#) to view the *5620 SAM-O XML Reference*.

Figure [7-1](#) shows the *5620 SAM-O XML Reference* with an XML sample object model.

Figure 7-1 Service object model



22014

7.2 Main menu

The main menu located at the top of the right panel provides hyperlinks to information in the following areas:

- Overview—lists all of the packages and their descriptions
- Package—provides information about the selected package and its information model diagram
- Class—provides information about the selected class
- Deprecated—lists all of the specifications that are deprecated for a specific 5620 SAM release
- Help—provides information about how to use and navigate the *5620 SAM-O XML Reference*

7.3 Package list

After you access the *5620 SAM-O XML Reference* (see Procedure 7-1), a table appears. The table includes the full package list and corresponding descriptions of all of the packages that are available using the XML interface. Click on a package link in the table for a general description of the package and the information model diagram.

The package list is also available from the upper left panel of the HTML document. When you choose All Classes/Types, or a specific package name from the list of packages, the content of the class and types list area in the lower left panel changes. Each package may contain classes, types, or both.

The information model is a UML drawing that shows the relationships amongst the classes in the selected package and other extended classes from which some properties, methods, and naming are inherited. The classes in the package are represented by white rectangles; the extended classes are represented by gray rectangles. When you click on one of the rectangles, the class information about the selected class is displayed.

Most packages have only one information model drawing. However, some of the larger packages (such as service, equipment, and sas) group member classes into separate information models that represent specific functional areas.

For example, the service package provides specifications for all of the areas that affect service creation, maintenance, configuration, and testing. As a result, the information model for the package is divided into more than one information model to address the following functional areas:

- Access Interface
- Connector and Composite Service
- GNE Info
- Base Service
- Logical Interface
- Operational Group and Encap Group
- STM Service Action
- STM Service Ping
- STM Service Trace
- SAP Policer and Policy Override

The \$core package, which is a root or base of all of the packages in the 5620 SAM-O, defines the base classes and types that all of the other classes use to inherit from. Perform Procedure 7-2 to view package information in the *5620 SAM-O XML Reference*.

7.4 Classes and types list

The class list in the lower left panel lists the classes and types that are defined in a package. The package name is displayed at the top of the list. Choose the package name to display an overview of the package in the right panel. Choose a class name or type to display information about each object in the right panel.

7.5 Class details

Each class has a page that may include the following information:

- [Description](#)
- [Inheritance hierarchy](#)
- [Direct subclass](#)
- [Parent hierarchy](#)
- [Children hierarchy](#)
- [Relationships](#)
- [Class type](#)
- [Methods returning children](#)
- [Naming](#)
- [Stats](#)
- [Properties](#)
- [Methods](#)
- [Supported network elements](#)

Description

The description provides information about the selected class. For example:

```
vpri
```

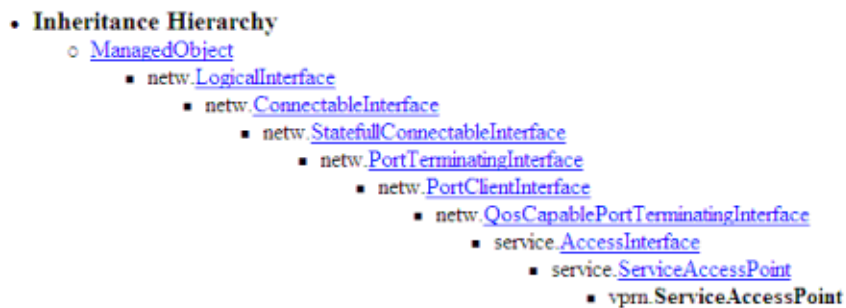
```
ServiceAccessPoint
```

```
A Service Access Point identifies the customer interface point in a
Routed CO VPRN solution.
```

Inheritance hierarchy

The inheritance hierarchy outlines the inheritance tree with the list of all classes that are extended to define a class, as shown in Figure 7-2.

Figure 7-2 Sample inheritance hierarchy



Direct subclass

The direct subclass lists all classes that inherit directly from the selected class, as shown in Figure 7-3.

Figure 7-3 Sample direct subclass

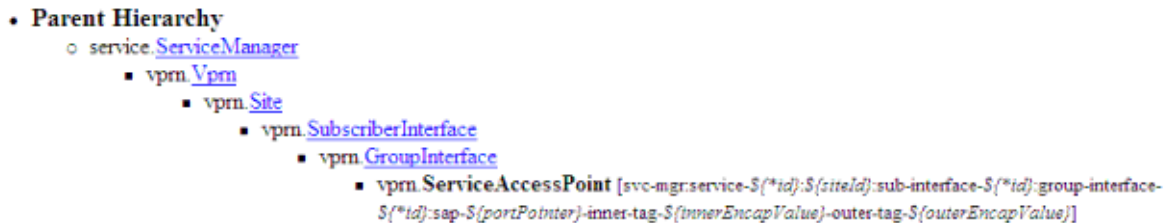
- **Direct Subclasses**
 - [vpri.MSap](#)

Parent hierarchy

The parent hierarchy indicates where an instance of the class may be located in the containment hierarchy, as shown in Figure 7-4.

The fully distinguished name, or objectFullName, specifies the 5620 SAM identifier for the specific object. See section 8.1 for more information about object full name.

Figure 7-4 Sample parent hierarchy



Children hierarchy

The children hierarchy lists the types of classes that may be contained in the hierarchy, for example, directly as children or indirectly as descendents, as shown in Figure 7-5.

Figure 7-5 Sample children hierarchy



Relationships

The format of relationships displays the relationship between the current class and other classes. The textual description is of the form:

```
<current-class> may be <relationship-description> <cardinality>
<other-class> (<relationship-type>)
```

where

<relationship-description> is one of the following: associated with, direct parent of, direct children of, contained under, or contain

<cardinality> describes one or many

<relationships-type> is one of the following: parent-child, child-parent, implied, ancestor-descendent, or descendent-ancestor

Figure 7-6 shows an example of relationships.

Figure 7-6 Sample relationships

• Relationships

- *vprn.Service.AccessPoint* may be associated with many *arp.ArpHosts* (implied relationship)
- one *vprn.Service.AccessPoint* may be a direct parent of many *clear.ClearRequests* (parent-child relationship)
- *vprn.Service.AccessPoint* may be associated with many *pppoe.PPPoESessions* (implied relationship)
- *vprn.Service.AccessPoint* may be associated with many *pppoe.PPPoESessions* (implied relationship)
- one *vprn.Service.AccessPoint* may be a direct parent of many *ressubscr.DbInfoSubscriberHosts* (parent-child relationship)
- one *vprn.Service.AccessPoint* may be a direct parent of many *ressubscr.HostTrackingInfoPerSaps* (parent-child relationship)
- one *vprn.Service.AccessPoint* may be a direct parent of one *ressubscr.SapSubMgmtCfg* (parent-child relationship)
- *vprn.Service.AccessPoint* may be associated with many *ressubscr.SubscriberHosts* (implied relationship)
- many *vprn.Service.AccessPoints* may be direct children of one *vprn.GroupInterface* (child-parent relationship)
- one *vprn.Service.AccessPoint* may be a direct parent of one *vprn.SapIcmpHostTracking* (parent-child relationship)
- many *vprn.Service.AccessPoints* may be contained under one *vprn.Site* (descendent-ancestor relationship)
- many *vprn.Service.AccessPoints* may be contained under one *vprn.Vprn* (descendent-ancestor relationship)

Class type

The class type is an abstract or a non-abstract class, which is defined as:

```
public [abstract] class name
```

Abstract classes are generally used to provide common property and method inheritance for concrete classes; for example, public abstract class *Vll*. The following is an example of a non-abstract class: public class *serviceAccessPoint*.

Naming

Objects are named by concatenating the parent and relative names. The relative name of an instance may be defined as: Relative Name=[*name*]. For example:

```
Relative Name=[svc-mgr]
```

Class properties present in the name are identified as: \${propertyName}. For example:

```
Relative Name=[ospf-${version}-instance-${instanceIndex}]
```

Auto-generated IDs are identified with an “*”, such as in \${*propertyName}.

Methods returning children

Some methods return children and some methods do not return children. These methods are denoted with the description; for example:

```
methodName() does not return children
```

For example, for netw.NetworkElement:

- configureChildInstanceWithResult() does not return children
- configureInstanceWithResult() does not return children
- findMultipleFilteredInstances() does not return children

Stats

The stats list the set of classes that monitor the class, including inherited statistics classes. Figure 7-7 shows an example.

Figure 7-7 Sample stats

Stats:
[arp.SapArpHostStats](#) [ressubscr.HostTrackStatsOnSap](#) [service.PppoeSapStats](#) [service.SapAtmPppStats](#)

Stats for

The Stats for section lists the set of classes that the class monitors, as shown in Figure 7-8.

Figure 7-8 Sample stats for

Stats For:
[vpls.AbstractL2AccessInterface](#) [ies.ServiceAccessPoint](#) [vprn.ServiceAccessPoint](#)

Supporting MIBs

The supporting MIBs section lists the MIB entries that the class reads. The listed MIB version reference entries are based on network element version, as shown in Figure 7-9.

Figure 7-9 Sample supporting MIBs

Supporting MIBs	
TIMETRA-SAP-MIB sapArpHostStatEntry	Alcatel-Lucent 7450 ESS 7.0 , 8.0 , 9.0 Alcatel-Lucent 7710 SR 7.0 , 8.0 , 9.0 Alcatel-Lucent 7750 SR 7.0 , 8.0 , 9.0 Alcatel-Lucent 7750 SR MG 2.0 , 3.0

Properties

The properties are listed alphabetically list of properties displayed for each class. The property definitions may include:

- comment
- type=type-specification— may be a fundamental type or link to another defined type. Children-Set indicates that objects of the specified type may be children of this class.
- default=value
- access=access-type—may be read-only, write-only, read-create, or read-write, which is the default. Read-create values can be set at creation time only.
- Mandatory on create (Indicates the value is required when an object is created)
- minimum=value
- maximum=value
- units=value
- SNMP Deploy/Resync: MIB.entry—defined for read-write access to the node
- Modification may shutdown object—changing the value may shut down the object. The GUI displays a dialog box for these items.
- Displayed (tab/group)=Displayed (tab/group)—displays the property name and tab location in the GUI. When a property does not have a tab or group information, the property is displayed in the general area.
- enums=enumeration—lists the enumerations when they are defined in a property
- bits—lists the bits when they are defined in a property

Figure 7-10 shows an example.

Figure 7-10 Sample properties

Properties	
clear.ClearRequest-Set	<i>type=Children-Set</i>
mcLagPropHoldTimeRemain	<p>"The value of mcLagPropHoldTimeRemain indicates the remaining time, in seconds, until MEPs on this SAP will react to a Multi-Chassis LAG protocol or port change. The value zero (0) indicates there are no pending events, or the SAP is not a MC-LAG SAP. This object corresponds to the global configuration timer: TIMETRA-IEEE8021-CFM-MIB:tmnxDot1agCfmMcLagPropHoldTime."</p> <p><i>type=long</i> <i>access=read-only</i> <i>Displayed(tab/group)=MC-LAG Hold Time (/Facility Meps)</i></p>
ressubscr.DbInfoSubscriberHost-Set	<i>type=Children-Set</i>
ressubscr.HostTrackingInfoPerSap-Set	<i>type=Children-Set</i>
ressubscr.SapSubMgmtCfg-Set	<i>type=Children-Set</i>
tunnelFaultNotification	<p>The value of tunnelFaultNotification specifies whether the SAP will 'accept' CFM fault notifications from a Tunnel MEP and process the notifications (i.e. do fault handling and/or fault propagation), or 'ignore' the notification. Both TIMETRA-SERV-MIB::svcEthCfmTunnelFaultNotification and this object MUST be set to 'accept' for the SAP to process the notification. The value 'notApplicable' is used by the system to represent a SAP which supports ETH-CFM, but not this object.</p> <p><i>type=service.TunnelFaultNotificationType</i> <i>default=accept</i> <i>Displayed(tab/group)=Tunnel Fault Notification (/Facility Meps)</i></p>
ypm.SapGrpHostTracking-Set	<i>type=Children-Set</i>

Overridden properties

A list of all properties that are overridden in this class. A property may be partly overridden and partly inherited through several classes.

Properties inherited from

Inherited and overridden properties may exist with links to definitions in other classes. Properties that are overridden by a specific class are displayed in italics. Figure 7-11 shows an example.

Figure 7-11 Sample properties inherited from

Properties inherited from service.ServiceAccessPoint
aaApplicationProfile aaGrpId aaPartId anep AnepStaticMap-Set antiSpoofing antispoof AntiSpoofingStaticHosts-Set callingStationId cpmProtEthCfmMonitorFlags <i>description</i> <i>displayedName</i> <i>id</i> <i>innerEncapValue</i> macMonitoring memberOperGroupName monitorOperGroupName <i>outerEncapValue</i> portPointer sapSubType service.OperGrpBindingMember-Set service.OperGrpBindingMonitor-Set

Methods

A list of methods is displayed for each class. The method definitions may include:

- comment
- Input Parameters: name : type [comment]

- Output Parameters: name : type [comment]
- Exceptions: name [comment]

Figure 7-12 shows an example.

Figure 7-12 Sample methods

Methods
<p>requestClearIgmpHostTracking</p> <p>Request clearing of all IGMP Host Tracking information for this ServiceAccessPoint. When this method returns resources for the clear have been allocated but the clear has not been performed yet. To retrieve the result eventually, use the returned request handle as input param of the retrieveClearRequest method of the clear.ClearCommandManager.</p> <p>Input Parameters:</p> <p>deployer : Deployer - the deployment state</p> <p>synchronousDeploy : boolean - (optional) Specify whether to block until the changes have been fully deployed to the network. A value of "true" means to block. A value of "false" means to return immediately. Default: false (asynchronous)</p> <p>clearOnDeployFailure : boolean - (optional) Specify whether to clear any failed deployers. A value of "true" means to clear. A value of "false" means to leave the failed deployer. Default: false</p> <p>deployRetries : int - (optional) The number of times to attempt re-deployment during synchronous deployment. This parameter is meaningless in the asynchronous case. Default: 0</p> <p>deployRetryInterval : long - (optional) The number of milliseconds to wait between deployment retries. This parameter is meaningless in the asynchronous case. Default: 0</p> <p>taskDescription : string - (optional) A user friendly description of what the operation does. This information will be used by the task manager.</p> <p>instanceFullName : string - the full name of the object.</p> <p>aInOnlyStats : boolean - Whether to clear the IGMP Host Tracking statistics only (host tracking info is not cleared).</p> <p>resultFilter : ResultFilter - (Optional) Filter for narrowing down the information returned per object</p> <p>continueOnFailure : continueOnFailure - (Optional) Continue processing requests in this stream if an exception occurs. Default: false</p> <p>Output Parameters:</p> <p>aOutResult : clear RequestHandleStruct - A request handle which allows the retrieval of the request (and its result when it becomes available) through the ClearCommandManager's retrieveClearRequest method.</p>

Methods inherited from

There may be a list of inherited methods with links to definitions in other classes. Figure 7-13 shows an example.

Figure 7-13 Sample methods inherited from

Methods inherited from service.AccessInterface
findSitesFor

Supported network elements

Supported network elements is a list of products and releases that support this class. Specifications may include:

- excluded chassis types chassis-list— not supported for the specified variants of this product.
- supported from version—supported for this product from the specified NE release and later. Multiple entries may be defined for different releases of the same product.
- supported for all releases—supported for all releases of the specified NE product.

If the supported network elements list does not appear, the class may be supported for any NE.

Figure 7-14 shows an example.

Figure 7-14 Sample supported network elements

Supported Network Elements	
Alcatel-Lucent 7750 SR	
Alcatel-Lucent 7710 SR	
Alcatel-Lucent 7450 ESS	<i>Excluded chassis types 7450-ESS1, 7450-ESS6, 7450-ESS6V</i> <i>Supported from 8.0 R1</i>
Alcatel-Lucent 7750 SR MG	

Product specifics

Product specifics is a list of properties which have different parameters for the NE vendor/versions. Property definitions may include:

- applicable=true or false—indicates whether the property is applicable for specific vendor/versions
- Excluded Chassis Types=chassis-list—not supported for the specified variants of this product.
- Supported from=version—supported for the product from the specified NE release or later. Multiple entries may be defined for different releases of the same product.
- default=value
- access=access type—indicates the vendor/version specific access type. Access type may be read-only, write-only, or read-create.
- write-access=yes or no—overrides the write access set using the access attribute in the base property
- minimum=value
- maximum=value
- suppressedEnums=enumeration—lists the enumerations that are suppressed for specific vendor/versions

7.6 Type details

Each package can also contain complex types. Each complex type has a page that contains the following list of values:

- Enumeration—includes name, value, display-name, whether selectable on GUI, and applicable list of classes
- BitMask—includes bit_name, value, display_name, and whether selectable on the GUI
- Struct—value definitions that include list of properties
- Deployer—specifies how and when deployment occurs
- DeploymentState—identifies whether the object is deployed or failed
- List—collection type, ordered collection of values, and duplicates allowed
- Pointer—object reference, expressed using a fully distinguished name
- Map—collection type, unordered key-value mapping, and duplicate keys are not allowed
- Set—collection type, unordered collection of values, and duplicates not allowed
- Children-Set—set of directly contained children for a specific object

Deployer, DeploymentState, and Children-Set are included in the \$score package and can be inherited from other classes.

7.7 XML schema changes

Deprecated properties and methods

This section lists all of the packages, properties, methods, classes, and enumerations that are deprecated starting from two releases before the current release. A list of hyperlinks is provided for all of the schema changes for releases from the 5620 SAM, Release 2.0 to the current release. Schema changes can include the update, removal, or addition of:

- packages
- objects (classes or types)
- methods
- properties

See Procedure 7-2 for more information about how to view schema changes between releases.



Note — If you are planning to use a 5620 SAM-O client that was tested against a previous release of the 5620 SAM, you must review your 5620 SAM-O usage with the list of schema changes before you start development, in order to identify changes that must be made to the packages, classes, types, methods, or properties that are used by the 5620 SAM-O client.

7.8 JMS changes

This section provides a list of hyperlinks for all of the JMS changes for the 5620 SAM, Release 5.0 or later. Each hyperlink opens a page that identifies changes to the JMS interface in that release. Changes are divided into two sections: event changes, and interface changes. See Procedure 7-3 for more information about how to view JMS changes between releases.



Note — If you are planning to use an existing 5620 SAM-O client that has been tested against a previous release of the 5620 SAM, you must review your 5620 SAM-O usage with the list of schema changes before you start development, in order to identify changes that must be made to packages, classes, types, methods, or properties that are used by the 5620 SAM-O client.

7.9 Procedures

Procedure 7-1 To display the *5620 SAM-O XML Reference*

- 1 Choose Help→User Documentation from the 5620 SAM main menu. A browser window opens to display the User Documentation Index page.
 - 2 Click on the *5620 SAM-O XML Reference* link.
 - 3 Click on a package name in the upper center panel.
 - 4 To view the relationship and inheritance drawing of the package, click on the package name in the lower center panel.
 - 5 To view class hierarchy, relationship, property, and supported NE information, click on the class name in the lower center panel.
-

Procedure 7-2 To view release-specific XML schema changes

- 1 Choose Help→User Documentation from the 5620 SAM main menu. A browser window opens to display the User Documentation Index page.
- 2 Click on the *5620 SAM-O XML Reference* link.

- 3 Scroll down to the 5620 SAM-O XML Schema Changes section in the right panel.
- 4 Click on the Schema Changes for Release *major_release_ID* link. The 5620 SAM-O XML Schema Changes page opens and displays information about the schema changes between the minor releases of the major release.



Warning — A schema change may affect an OSS application. After a 5620 SAM system upgrade, you must ensure that each OSS application complies with the current schema before you deploy the application, otherwise serious system or network damage may result.

Procedure 7-3 To view release-specific JMS changes

- 1 Choose Help→User Documentation from the 5620 SAM main menu. A browser window opens to display the User Documentation Index page.
 - 2 Click on the *5620 SAM-O XML Reference* link.
 - 3 Scroll down to the 5620 SAM-O JMS Changes section in the right panel.
 - 4 Click on the JMS Changes for Release *major_release_ID* link. The 5620 SAM-O JMS Changes page opens and displays information about changes between the minor releases of the major release.
-

Procedure 7-4 To view deprecated schema items

- 1 Choose Help→User Documentation from the 5620 SAM main menu. A browser window opens to display the User Documentation Index page.
- 2 Click on the *5620 SAM-O XML Reference* link.
- 3 Click on the deprecated properties link in the upper center panel. The Deprecated Packages / Properties / Methods / Classes / Enumerations page opens and displays a list of the deprecated items.



Warning — A deprecation may affect an OSS application. After a 5620 SAM system upgrade, you must ensure that each OSS application complies with the current schema before you deploy the application, otherwise serious system or network damage may result.

Procedure 7-5 To view the general methods and types

- 1 Choose Help→User Documentation from the 5620 SAM main menu. A browser window opens to display the User Documentation Index page.
 - 2 Click on the *5620 SAM-O XML Reference* link.
 - 3 Click on the general methods/types link in the upper center panel. The General Methods / Types page opens and displays a list of the methods and types that are for general use.
-

Procedure 7-6 To view the supported device types

- 1 Choose Help→User Documentation from the 5620 SAM main menu. A browser window opens to display the User Documentation Index page.
 - 2 Click on the *5620 SAM-O XML Reference* link.
 - 3 Click on the supported products link in the upper center panel. The General Methods / Types page opens and displays a list of the supported device types for the current release.
-

8 — 5620 SAM-O XML Schema

8.1 5620 SAM-O XML Schema overview 8-2

8.1 5620 SAM-O XML Schema overview

An XML schema describes the structure and allowed elements of a document. An XML schema document has a .xsd extension and contains schema definitions. You can use an XML schema to do the following:

- Determine the valid elements and parameters, and the data types for the elements.
- Create and enforce the rules for the number and order of child elements.
- Determine default and fixed values for elements and parameters.

Procedure 8-1 To display specific 5620 SAM-O XML package information

- 1 Choose Help→User Documentation from the 5620 SAM main menu. A browser window opens to display the User Documentation Index page.
- 2 Click on the 5620 SAM-O XML Schema link. The All 5620 SAM OSS XML Schema page is displayed. The page lists each 5620 SAM-O *packageMethods.xsd* and *packageTypes.xsd* file

where *package* is the name of a 5620 SAM-O package

The list also includes standard XML-package schemas such as *xmlApiTypes.xsd* and *xmlApiMethods.xsd*, and specific XML-package schemas.

- 3 Click on an XML schema document link. The package information is displayed.

XML standard and specific schema files

Standard and specific XML schema definitions are available from the *xmlApiTypes.xsd* and *xmlApi.xsd* files. Standard and specific XML methods are available from the *xmlApiMethods.xsd* file.

The *xmlApi.xsd* defines and lists all of the *PackageName.xsd* schema files and the standard XML xsd schema definitions.

The *xmlApiTypes.xsd* defines all of the XML API standard types from simple types like Date, Char, and IPAddress, to more complex structures like Filters, Exceptions, and Any types.



Note — General types *Filterholder*, *ChildrenFilterHolder*, and *ResultFilter* are also defined in the *5620 SAM-O XML Reference* general methods and types. See chapter 7 for information about viewing the general methods and types.

The *xmlApiMethods.xsd* defines the core methods that a 5620 SAM-O client can use to perform specific actions. These core methods are also described in the *5620 SAM-O XML Reference*. See chapter 7 for information about displaying the *5620 SAM-O XML Reference*.

Table 8-1 describes the core methods that a 5620 SAM-O client can use to perform specific actions. These methods are formally defined in *xmlApiMethods.xsd*.

Table 8-1 Core XML methods

Method	Purpose	Input parameters	Output parameters
deregisterLogToFile	To stop the creation of accounting statistics files	<ul style="list-style-type: none"> fullClassName (optional) Package qualified class name in dot-separated format. If this parameter is not specific then it deregisters for all accounting statistics (string) jmsClientId The JMS Client ID (string) 	—
deregisterSasLogToFile	To stop the creation of OAM test results file	<ul style="list-style-type: none"> fullClassName (optional) Package qualified class name in dot-separated format. If this parameter is not specific then it deregisters for all types of test results (string) jmsClientId The JMS Client ID (string) continueOnFailure (optional) Continue processing requests in this stream if an exception occurs. The default is false. (Boolean) 	—
deregisterNotification	To deregister an XML filter and stop advanced JMS message filtering for the OSS client	<ul style="list-style-type: none"> jmsClientId The JMS Client ID (string) 	—
find	To return the set of objects of the specified type which match the specified filter criteria	<ul style="list-style-type: none"> fullClassName Package qualified class name in dot-separated format filter (strongly recommended) Filter which objects are returned based on the properties of the class specified by fullClassName timeout (optional) Specifies the time, in milliseconds, after which the find operation times out. The request is aborted and an exception is returned. resultFilter (strongly recommended) Filter which attributes are returned for each object 	<ul style="list-style-type: none"> result (generic.CommonManagedEntityInformation) The list of objects which match the given filter criteria.

(1 of 4)

Method	Purpose	Input parameters	Output parameters
findToFile	To find the set of objects of the specified type which match the given filter criteria, and places the results in the specified file on the server. Upon completion of the command, a fileAvailable event is generated.	<ul style="list-style-type: none"> • fullClassName Package qualified class name in dot-separated format • fileName The relative file in which to store the results of this find operation. The format is: [s]ftp://[user:password@]host[:port]/directory/file The following rules must be applied to the format: <ul style="list-style-type: none"> • When using a remote client, the directory must be relative to the default FTP directory for the remote client. • The characters '/' and ';' are reserved and must be encoded. For example, to specify a file in /tmp directory on the remote client, the path must be encoded as ftp://name@password/%2Ftmp/filename.xml. The path ftp://name@password/tmp/filename.xml specifies a file in a /tmp directory under the default FTP directory of the user. See RFC 1738 Uniform Resource Locators (URL), Section 3.2.2 for more information about the FTP URL path requirements. • timeStamp (optional) Includes the timestamp into the filename of the saved file • timeout (optional) Specifies the time, in milliseconds, after which the findToFile operation times out. The request is aborted and an exception is returned. The file may be partially written and may not be valid. An exception indicating a timeout has occurred will be present in the JMS FileAvailableEvent also. • filter (strongly recommended) Filter on properties of the class corresponding to fullClassName • resultFilter (strongly recommended) Filter for narrowing down the information returned per object • synchronous (optional) Specifies that the method is run synchronously and that the result is returned only when the find is complete. The default is true. 	—
ping	To check the connectivity of the 5620 SAM server.	—	—

(2 of 4)

Method	Purpose	Input parameters	Output parameters
registerLogToFile	To create accounting stats file based on the specified class type. Accounting stats Files are created and are placed in the specified directory on the server. Each time a file is created, a LogFileAvailableEvent event is generated. If there is no JMS client subscribed within a given time, the 5620 SAM server deregisters the request.	<ul style="list-style-type: none"> • fullClassName Package qualified class name in dot separated format. A comma separated list of accounting statistics classes (string) • dirName The relative path to the subdirectory where the files are to be saved. The path is relative to the OSS XML export directory/accountingStats. The use of separate directories for different applications that export statistics is recommended. • compress Specifies whether export files should be compressed when they are created (optional) • jmsClientId The JMS Client ID that is notified when a file is created (string) • resultFilter Specifies the attributes that are to be included in the exported data records (optional) 	—
registerSasLogToFile	To create OAM test results file based on class type that is specified. Files are created and are placed in the specified directory on the server. Every time that a file is created, a LogFileAvailableEvent event is generated. If there is no JMS client subscribed within some time, the 5620 SAM server deregisters the request.	<ul style="list-style-type: none"> • fullClassName Package qualified class name in dot separated format. A comma separated list of sas.TestResult classes. Superclasses may be specified. (string) • dirName The relative path to the subdirectory where the files are to be saved. The path is relative to the OSS XML export directory/oam. The field is mandatory. The use of separate directories for different applications that export different test results or use different filters is recommended. • compress Specifies whether export files should be compressed when they are created. A value of True saves files with gzip compression and a filename extension of gz. The default is False.(optional) • jmsClientId The JMS Client ID that is notified when a file is created (string) • resultFilter Specifies the attributes that are to be included in the exported data records (optional) • continueOnFailure (optional) Continue processing requests in this stream if an exception occurs. The default is False. (Boolean) 	—

(3 of 4)

Method	Purpose	Input parameters	Output parameters
registerNotification	To register an advanced JMS message filter and begin using the filter for messages sent to the OSS client	<ul style="list-style-type: none"> • jmsClientId The JMS Client ID (string) • List of filter criteria, in XML format; see chapter 4 for information about advanced JMS message filtering 	—
sleep	To halt request processing for the given period of time (in milliseconds) before proceeding to the next request element.	<ul style="list-style-type: none"> • milliseconds The length of time in milliseconds to wait before executing the next request element. (long) 	—
timeStamp	To view the current Java time on the 5620 SAM server, in milliseconds. The execution time of a command or group of commands can be determined by surrounding them with timeStamp commands and taking the delta of their return values.	<ul style="list-style-type: none"> • includeFormatted (optional) Include formatted timestamp in the response (Boolean) 	<ul style="list-style-type: none"> • millis The difference in milliseconds between the current time and midnight, Jan 1, 1970 UTC (long) • timeStamp The formatted time stamp (string)
version	To return the current 5620 SAM software version	—	<ul style="list-style-type: none"> • version The version of the server (string) • baseVersion The major.minor version of the server (string) • build The build of the server (string) • patch The patch level of the server (string)

(4 of 4)

Other XML schema definitions that are specific to a 5620 SAM function are contained in the following files:

- **<xx>Methods.xsd** — files contain the methods that are associated with the packages
- **<xx>Types.xsd** — files contain the objects (classes), information structures, bitmasks, and enumerations that are associated with the packages where **<xx>** is the package type

For example, one of the packages listed in section 6.2 is the fm package. The fm package functionality is described in the fmTypes.xsd file and fmMethods.xsd files.

See the W3C websites listed in the Preface for more general information about XML schemas.

XML package methods schema files

XML methods are defined in PackageNameMethods.xsd files. All method definitions begin with a header containing:

- method name
- function — one of MODIFIER, RETRIEVER
- scope — one of Instance (need to supply an instanceFullName), Class (need to supply a className)
- input parameters
- output parameters
- exceptions

Code 8-1 is an example of a method definition header for the clearFaults method of the FaultManager class, which is included in the fm package.

Code 8-1: Sample method definition header of the fm.FaultManager.clearFaults method

```
<!--
= fm.FaultManager.clearFaults
= [MODIFIER]
= [SCOPE: CLASS]
==
= [IN]
= deployer - Deployer
= The deployment state.
=
= synchronousDeploy - xsd:boolean
= (optional) Specify whether to block until the changes have been
= fully deployed to network. A value of "true" means to block.
= A value of "false" means to return immediately.
= Default: false (asynchronous)
=
= clearOnDeployFailure - xsd:boolean
= (optional) Specify whether clear the deployer if a failure occurs.
= A value of "true" means to clear.
= A value of "false" means to leave the failed deployer.
= Default: false
=
= deployRetries - xsd:int
= (optional) The number of times to attempt re-deployment during
= synchronous deployment. This parameter is meaningless in the
= asynchronous case.
= Default: 0
=
= deployRetryInterval - xsd:long
= (optional) The number of milliseconds to wait between deployment
= retries. This parameter is meaningless in the asynchronous case.
= Default: 0
=
= baseInstanceFullName - xsd:string
= alarmableInstanceFilter - FilterHolder
= faultFilter - FilterHolder
= scopeType - xsd:int
= scopeDepth - xsd:int
=
```

```

= [OUT]
=
= [EXCEPTIONS]
= fm.FaultManager.clearFaultsException
-->

```

Common elements of methods

Table 8-2 describes elements that are common to all 5620 SAM methods.

Table 8-2 Common elements of methods

Element	Description	Options
deployer	Specifies the deployment state.	—
synchronousDeploy	Specifies whether synchronous or asynchronous deployment is used.	<ul style="list-style-type: none"> false (default) Asynchronous deployment true Synchronous deployment
clearOnDeployFailure	Specifies whether to clear the deployer if a failure occurs.	<ul style="list-style-type: none"> false (default) The deployer is not cleared true (recommended) Clear the deployer
deployRetries	Specifies the number of times that re-deployment is attempted during a synchronous deployment. This is an optional parameter.	0 (default and recommended)
deployRetryInterval	Specifies the time, in milliseconds, between deployment attempts in synchronous deployments. This parameter is optional.	0 (default)
instanceFullName	Specifies the full name of the object.	—
configInfo	Specifies any class type.	

XML package types schema files

XML object classes are defined in *PackageNameTypes.xsd* files. *PackageName* is the name of the package. All object class definitions begin with a header containing:

- Name
- Type - class
- Abstract - yes/no
- Singleton - yes/not
- Category - one of service, topology, general
- Deployable - yes/no
- Read access - list of users or all

- Write access - list of users or all
- Inheritance Hierarchy - inheritance tree

Code 8-2 is an example of an Object Class Type definition header for the AdditionalTextAttribute class that is included in the fm package.

Code 8-2: Sample Object Class definition header

```
<!--
=
= Name: fm.AdditionalTextAttribute
= Type: Class
= Abstract: no
= Singleton: no
=
= Category: fault
= Deployable: no
= Read Access: all
= Write Access: admin, operations, fault
=
= Inheritance Hierarchy:
=
= <root>
= |
= fm.AdditionalTextAttribute
=
-->
```

Common elements of classes (types)

All object classes defined in the XML Schema definition files contain the following standard elements:

- actionMask — specifies the operation to occur on an object during configuration
- children-Set — set of contained children for the specific object
- objectFullName — specifies the 5620 SAM identifier of the instance of the object
- selfAlarmed — specifies if this object raises alarms

Code 8-3 provides an example of the XML schema definition for these standard properties for the fm.AdditionalTextAttribute class.

Code 8-3: Standard properties definition for a class example

```
<xsd:complexType name="fm.AdditionalTextAttribute">
  <xsd:all>
<!-- standard properties -->
<xsd:element name="objectFullName" type="xsd:string" minOccurs="0"/>
<xsd:element name="actionMask" type="generic.ModifierActionMask"
minOccurs="0"/>
<xsd:element name="children-Set" type="Children-Set" minOccurs="0"/>
<xsd:element name="selfAlarmed" type="xsd:boolean" minOccurs="0"/>
<!--
```

actionMask

Every class definition contains an <actionMask> element. The <actionMask> specifies what operation occurs to the object during configuration, specific to each configured object. The <actionMask> is only used during configuration. You can use the genericTypes.xsd file to specify the definition of the <actionMask> (generic.ModifierActionMask). Table 8-3 lists the valid action types and the numeric equivalents.

Table 8-3 Action types

Action Type	Numerical equivalent	Description
create	1	To create an object based on the elements (parameters) received from the XML API. Only read-write and read-create attributes are allowed. The read-create attribute is mandatory.
modify	2	To modify an existing object based on the elements (parameters) received from the XML API. Only read-write attributes are allowed.
delete	4	To delete an existing object.
reset	8	To reset the elements (parameters) of an existing object to the default values.



Note — Not all actionMasks are applicable to all classes.

Alcatel-Lucent recommends that you use one actionType for each request. However, you can combine the actionTypes. For example, when you want to create and modify an object that does not exist, you can specify the create and modify bits in the same request. Code 8-4 shows an example of a combined request. Alternately, you can use the numeric value of the bit instead of the text.

Code 8-4: Combined create and modify request

```
<actionMask>
<bit>create</bit>
<bit>modify</bit>
</actionMask>
```

children-Set

The objects in the 5620 SAM-O model are organized in an object hierarchy. The specific parent-child relationships in the hierarchy are captured by the Children-Set property for a given parent object.

Code 8-5 is an example that shows how the children-Set property is used in an XML request to configure the Base Card setting the card slot to In Service, for slotId 7 and configuring the card to a “card_iom2_20g card.

Code 8-5: Children-Set example

```
<equipment.CardSlot>
```

```

    <actionMask>
      <bit>modify</bit>
    </actionMask>
    <children-Set>
      <equipment.BaseCard>
        <actionMask>
          <administrativeState>inService</administrativeState>
          <slotId>7</slotId>
          <specificType>
            <bit>card_iom2_20g</bit>
          </specificType>
        </equipment.BaseCard>
      </children-Set>
    </equipment.CardSlot>

```

Object full name

The <objectFullName> element specifies the 5620 SAM unique identifier of the object instance. This name is unique across the entire 5620 SAM system. The object full name is also referred to as the object fully distinguished name. The format of the object full name depends on the type of object. Table 8-4 lists a sample of the object types and how they are uniquely identified.



Note — Do not compose or parse the <objectFullName> element. You can only use the <objectFullName> element in a complete format as a unique identifier or key. The element identifies or retrieves a specific 5620 SAM object.

You must use the objectFullName format when you define an <x>ObjectPointer element where <x> defines the object. For example:

```

<ingressPolicyObjectPointer>

Access Ingress:1

</ingressPolicyObjectPointer>

```

Table 8-4 Sample unique identifiers for network objects

Network object (class)	Format	Example
Network element		
Network element	Created for each managed node and forms the base of site-specific objects. <i>network:sitelD</i>	network:10.1.1.88
Policies		
Policy manager	A policy manager exists for each type of policy at the network and element level and contains sets of policies of a type. For example, access and slope. <i>policy Manager Type</i> for network level or <i>network:sitelD:policyManagerType</i> for element level	Access Egress or network:10.1.1.88:Access Egress
Access egress policy (aengr.Policy)	<i>policy Manager Type:ID</i> or <i>network:sitelD:policyManagerType:policyID</i> for element level	Access Egress:2 or network:10.1.1.88:Access Egress:2
Access egress queue (aengr.Queue)	<i>network:sitelD:policyManagerType:policyID:queue-ID</i>	network:10.1.1.88:Access Egress:2:queue-1
Access ingress policy (aingr.Policy)	<i>policyManagerType:ID</i> or <i>network:sitelD:policyManagerType:policyID</i> for element level	Access Ingress:2 or network:10.1.1.88:Access Ingress:2
Access ingress forwarding class (aingr.ForwardingClass)	<i>policyManagerType:forwarding-class-type</i> or <i>network:sitelD:policyManagerType:forwarding-class-type</i> for element level	Access Egress:2:forwarding-class-be or network:10.1.202.94:Access Egress:2:forwarding-class-be
Access ingress queue (aingr.Queue)	<i>network:sitelD:policyManagerType:policyID:queue-ID</i>	network:10.1.1.88:Access Ingress:2:queue-1
Network policy (niegr.Policy)	<i>Network:ID</i> or <i>network:sitelD:Network:ID</i> for element level	Network:2 network:10.1.1.88:Network:2
Network forwarding class (niegr.ForwardingClass)	<i>NetworkID:forwarding-class-type</i> or <i>network:sitelD:NetworkID:forwarding-class-type</i> for element level	network:10.1.1.88:Network:3:forwarding-class-af
File policy (file.Policy)	<i>File:policyID</i> or <i>network:sitelD:File:ID</i> for element level	File:2 or network:10.1.1.88:File:2
Slope policy (slope.Policy)	<i>slopeName</i> or <i>network:sitelD:Slope:default</i>	network:10.1.1.88:Slope:default

(1 of 3)

Network object (class)	Format	Example
IP filter policy (aclFilter.IPFILTER)	IP Filter: <i>policyID</i> or <i>network:siteID</i> :IP Filter: <i>policyID</i> for element level	IP Filter:5 or network:10.1.1.88:IP Filter:5
Mediation security policy (security.MediationPolicy)	pollerManager:mediation-security- <i>policyID</i>	pollerManager:mediation-security-1
Deployer policy (mediation.DeploymentPolicy)	Default.DeployerBank:Policy	Default.DeployerBank:Policy
Deployer (generic.DeployerInfo)	Default.DeployerBank:depl- <i>ID</i>	Default.DeployerBank:depl-322
Router		
Router (rtr.VirtualRouter)	There is only one virtual router for each node. <i>network:siteID</i> :router-1 for element level	network:10.1.1.88:router-1
Network Interface (rtr.NetworkInterface)	<i>network:siteID</i> :router-1:ip-interface- <i>ID</i> for element level	network:10.1.1.88:router-1:ip-interface-2
Static route (rtr.StaticRoute)	<i>network:siteID</i> :router-1:SR- <i>ID</i> - <i>DestinationsitelD</i> - <i>IP mask</i> for element level	network:10.1.1.88:router-1:SR-2-12.119.60.0-255.255.255.0
OSPF		
OSPF (ospf.area)	ospf:area- <i>siteID</i>	ospf:area-2.2.2.2
OSPF site (ospf.Site)	<i>network:siteID</i> :router-1:ospf for element level	network:10.11.1.219:router-1:ospf
OSPF area (ospf.AreaSite)	<i>network:siteID</i> :router-1:ospf:areaSite- <i>siteID</i> for element level	network:10.11.1.219:router-1:ospf:areaSite-0.0.0.0
OSPF interface (ospf.interface)	<i>network:siteID</i> :router-1:ospf:areaSite- <i>siteID</i> :interface- <i>siteID</i> for element level	network:10.11.1.219:router-1:ospf:areaSite-0.0.0.0:interface-3.3.3.3.0
OSPF export policy (ospf.ExportPolicy)	<i>network:siteID</i> :router-1:ospf:exportPolicy for element level	network:10.11.1.219:router-1:ospf:exportPolicy
BGP		
BGP (bpg.Site)	<i>network:siteID</i> :router-1:bpg for element level	network:10.11.1.219:router-1:bpg
BGP peer group (bpg.PeerGroup)	<i>network:siteID</i> :router-1:bpg:group- <i>Group name</i> for element level	network:10.11.1.219:router-1:bpg:group-Sample-Client
BGP group import policy (bpg.GroupImportPolicy)	<i>network:siteID</i> :router-1:bpg:group- <i>Group name</i> :importPolicy for element level	network:10.11.1.219:router-1:bpg:group-Sample-Client:importPolicy
BGP group export policy (bpg.GroupExportPolicy)	<i>network:siteID</i> :router-1:bpg:group- <i>Group name</i> :exportPolicy for element level	network:10.11.1.219:router-1:bpg:group-Sample-Client:exportPolicy
BGP site export policy (bpg.SiteExportPolicy)	<i>network:siteID</i> :router-1:bpg:exportPolicy for element level	network:10.11.1.219:router-1:bpg:exportPolicy
BGP site import policy (bpg.SiteExportPolicy)	<i>network:siteID</i> :router-1:bpg:importPolicy for element level	network:10.11.1.219:router-1:bpg:importPolicy
BGP confederation (bpg.Confederation)	<i>network:siteID</i> :router-1:confederation- <i>ID</i> for element level	network:10.11.1.219:router-1:confederation-100
BGP confederation member (bpg.ConfederationMember)	<i>network:siteID</i> :router-1:confederation- <i>ID</i> :member- <i>ID</i> for element level	network:10.11.1.219:router-1:confederation-100:member-10
LDP		

(2 of 3)

Network object (class)	Format	Example
LDP (ldp.Site)	<i>network:siteID:router-1:ldp</i> for element level	network:10.1.1.219:router-1:ldp
MPLS		
MPLS (mpls.Site)	<i>network:siteID:router-1:mpls</i> for element level	network:10.1.1.219:router-1:mpls
MPLS interface (mpls.interface)	<i>network:siteID:router-1:mpls:interface-ID</i> for element level	network:10.1.1.219:router-1:mpls:interface-1
MPLS dynamic LSP (mpls.DynamicLSP)	<i>lsp:from-Source siteID-id-ID</i>	lsp:from-10.1.1.88-id-1
MPLS LSP path (mpls.LspPath)	<i>lsp:from-Source siteID-id-ID:lsppath-ID</i>	lsp:from-10.1.1.88-id-1:lsppath-2
MPLS provisioned path (mpls.ProvisionedPath)	<i>provisionedMplsTePath:from-Source siteID-id-ID</i>	provisionedMplsTePath:from-10.1.1.88-id-1
MPLS provisioned hop (mpls.ProvisionedHop)	<i>provisionedMplsTePath:from-Source siteID-id-ID:hop-ID</i>	provisionedMplsTePath:from-10.1.1.88-id-1:hop-2
MPLS tunnel (mpls.Tunnel)	<i>mplsTunnel:from-Source siteID-id-ID</i>	mplsTunnel:from-10.1.1.88-id-1
Subscribers		
Subscriber (subscr.Subscriber)	<i>subscriber:ID</i>	subscriber:21
Services		
Service (service.Service)	<i>serviceManager:service:ID</i>	svc-mgr:service-12
Service site (service.Site)	<i>serviceManager:service:ID:siteID</i>	svc-mgr:service-5:10.1.1.91
Service access interface (service.AccessInterface)	<i>serviceManager:serviceID:siteID:ip-interface-ID</i>	svc-mgr:service-5:10.1.1.91:interface-1/1/4-inner-tag-0-outer-tag-11 or svc-mgr:service-52:10.1.202.93:ip-interface-2051
Channels		
STS192 Channel (sonetequipment.Sts192Channel)	<i>network:siteID:shelf-ID:cardSlot-ID:card:daughterCardSlot-ID:daughterCard:port-ID:Sts192Channel</i>	network:10.1.1.218:shelf-1:cardSlot-2:card:daughterCardSlot-1:daughterCard:port-1:sts192-1
DS3/E3 Channel (tdmequipment.DS3E3Channel)	<i>network:siteID:shelf-ID:cardSlot-ID:card:daughterCardSlot-ID:daughterCard:port-ID:DS3E3Channel-ID</i>	network:10.1.1.218:shelf-1:cardSlot-3:card:daughterCardSlot-1:daughterCard:port-1:ds3e3-1
DS0 Channel Group (tdmequipment.DS0ChannelGroup)	<i>network:siteID:shelf-ID:cardSlot-ID:card:daughterCardSlot-ID:daughterCard:port-ID:DS1E1Channel-ID:DS0ChannelGroup-ID</i>	network:10.1.1.218:shelf-1:cardSlot-3:card:daughterCardSlot-1:daughterCard:port-1:ds1e1-3:ds0Grp-3

(3 of 3)



Note 1 — There is no documented list of object full name formats. The samples in Table 8-4 are not guaranteed to remain valid from release to release of the 5620 SAM. It is recommended that 5620 SAM-O XML request writers find FDNs to use in their requests by performing a <find> request for the class of the object they are looking for. See section 12.2 for more information about the <find> method.

For classes with no inventory present, the find will not return a result where you can view an object full name. Create the object using the 5620 SAM, assuming the physical network element is configured to allow the creation of the object. You can then perform the find request that allows you to view the object full name for the created object.

You can also monitor the 5620 SAM logs for the FDN when you create or modify an object using the 5620 SAM GUI. See section 9.1 for more information about log files.

Note 2 — Do not compose or parse the <objectFullName> element. You can only use the <objectFullName> element in a complete format as a unique identifier or key. The element identifies or retrieves a specific 5620 SAM object.

Element definitions

Elements that are defined in the XSD files can be included in a class definition and typically follow the standard properties or they can exist on their own. The element definitions begin with a header detailing:

- Name — element name
- Defined in — class element belongs to
- Min/Max — min/max values
- Default — default value
- Access — one of Read-Only, Read-Create, Read-Write

Code 8-6 provides an example of the XML schema definition for an element definition for the fm.AdditionalTextAttribute.order element, which is part of the fm.AdditionalTextAttribute class.

Code 8-6: Sample definition for an element example

```
<!--
- Name: order
- Defined in: fm.AdditionalTextAttribute
- Min/Max: 0 to 65535
- Default: 0
- Access: Read-Write
-->
<xsd:element name="order" type="xsd:int" minOccurs="0"/>
```

Accessibility types

Each element (parameter) has an accessibility type. You can view the accessibility type in the *PackageNameTypes.xsd* file, where *PackageName* is the name of the 5620 SAM-O package type. The accessibility types are:

- read-only, which specifies that the element is read-only and cannot be used in a configuration request
- read-write, which specifies that the element is read-write at any time using the create or modify actionTypes
- read-create, which specifies that the element can only be written during a create actionType request

Table 8-5 describes codes 8-7 through 8-9, which show sample XML schema definitions.

Table 8-5 XML schema definition samples

Sample	Description	File
Code 8-7	Sample XML definition for a class	fmTypes.xsd
Code 8-8	Sample XML definition for an element	fmTypes.xsd
Code 8-9	Sample XML definition for a method	fmMethods.xsd

Code 8-7: Sample XML schema definition for a class

```
<!--
=
= Name:          fm.AdditionalTextAttribute
= Type:          Class
= Abstract:      no
= Singleton:     no
=
= Category:      fault
= Deployable:    no
= Read Access:   all
= Write Access:  admin, operations, fault
=
= Inheritance Hierarchy:
=
= <root>
= |
= fm.AdditionalTextAttribute
=
-->
<xsd:complexType name="fm.AdditionalTextAttribute">
  <xsd:all>
    <!-- standard properties -->
    <xsd:element name="objectFullName" type="xsd:string" minOccurs="0"/>
    <xsd:element name="actionMask" type="generic.ModifierActionMask"
minOccurs="0"/>
    <xsd:element name="children-Set" type="Children-Set" minOccurs="0"/>
    <xsd:element name="selfAlarmed" type="xsd:boolean" minOccurs="0"/>
  </xsd:all>
</xsd:complexType>
```

```

<!--
    - Name:      attributeName
    - Defined in: fm.AdditionalTextAttribute
    - Default:
    - Access:    Read-Create
    -->
    <xsd:element name="attributeName" type="xsd:string" minOccurs="0"/>
<!--
    - Name:      order
    - Defined in: fm.AdditionalTextAttribute
    - Min/Max:    0 to 65535
    - Default:    0
    - Access:    Read-Write
    -->
    <xsd:element name="order" type="xsd:int" minOccurs="0"/>
    <!-- Direct children -->
    <!-- Read only common properties -->
    <xsd:element name="name" type="xsd:string" minOccurs="0"/>
    <xsd:element name="deploymentState" type="DeploymentState" minOccurs="0"/>
  </xsd:all>
</xsd:complexType>
<xsd:element name="fm.AdditionalTextAttribute" type="fm.AdditionalTextAttribute"/>
<xsd:element name="fm.AdditionalTextAttribute-Set" type="Children-Set"/>
<!--
=
= Name:      fm.AdditionalTextPolicy
= Type:      Class
= Abstract:   no
= Singleton: no
=
= Category:   fault
= Deployable: no
= Read Access: all
= Write Access: admin, operations, fault
=
= Inheritance Hierarchy:
=
= <root>
= |
= fm.AdditionalTextPolicy
=
-->
<xsd:complexType name="fm.AdditionalTextPolicy">
  <xsd:all>
    <!-- standard properties -->
    <xsd:element name="objectFullName" type="xsd:string" minOccurs="0"/>
    <xsd:element name="actionMask" type="generic.ModifierActionMask"
minOccurs="0"/>
    <xsd:element name="children-Set" type="Children-Set" minOccurs="0"/>
    <xsd:element name="selfAlarmed" type="xsd:boolean" minOccurs="0"/>
    <!--
    - Name:      domain
    - Defined in: fm.AdditionalTextPolicy
    - Default:
    - Access:    Read-Create
    -->
    <xsd:element name="domain" type="xsd:string" minOccurs="0"/>
    <!--
    - Name:      objectType
    - Defined in: fm.AdditionalTextPolicy
    - Default:
    - Access:    Read-Create
    -->
    <xsd:element name="objectType" type="xsd:string" minOccurs="0"/>
    <!--
    - Name:      description
    - Defined in: fm.AdditionalTextPolicy
    - Min/Max:    0 to 80
    - Default:
    - Access:    Read-Write

```

```
-->
<xsd:element name="description" type="xsd:string" minOccurs="0"/>
<!--
-   Name:          overwrite
-   Defined in:    fm.AdditionalTextPolicy
-   Default:       false
-   Access:        Read-Write
-->
<xsd:element name="overwrite" type="xsd:boolean" minOccurs="0"/>
```

Code 8-8: Sample XML schema definition for an element

```
<!-- =
=   Name: fm.Severity
=   Type: Enumeration
=
-->

<xsd:simpleType name="fm.Severity">
  <xsd:restriction base="xsd:string">
    <!--
    -   noalarm (-1)
    -
    -   Applicable Classes:
    -       any class
    -->
    <xsd:enumeration value="noalarm"/>
    <xsd:enumeration value="-1"/>
    <!--
    -   cleared (1)
    -
    -   Applicable Classes:
    -       any class
    -->
    <xsd:enumeration value="cleared"/>
    <xsd:enumeration value="1"/>
    <!--
    -   indeterminate (2)
    -
    -   Applicable Classes:
    -       any class
    -->
    <xsd:enumeration value="indeterminate"/>
    <xsd:enumeration value="2"/>
    <!--
    -   info (3)
    -
    -   Applicable Classes:
    -       any class
    -->
    <xsd:enumeration value="info"/>
    <xsd:enumeration value="3"/>
    <!--
    -   condition (4)
    -
    -   Applicable Classes:
    -       any class
    -->
    <xsd:enumeration value="condition"/>
    <xsd:enumeration value="4"/>
    <!--
    -   warning (5)
    -
    -   Applicable Classes:
    -       any class
    -->
    <xsd:enumeration value="warning"/>
    <xsd:enumeration value="5"/>
    <!--
    -   minor (6)
```

```

-
-   Applicable Classes:
-       any class
-->
<xsd:enumeration value="minor"/>
<xsd:enumeration value="6"/>
<!--
-   major (7)
-
-   Applicable Classes:
-       any class
-->
<xsd:enumeration value="major"/>
<xsd:enumeration value="7"/>
<!--
-   critical (8)
-
-   Applicable Classes:
-       any class
-->
<xsd:enumeration value="critical"/>
<xsd:enumeration value="8"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:element name="fm.Severity" type="fm.Severity"/>

```

Code 8-9: Sample XML schema definition for a method

```

<!--
= fm.FaultManager.clearFaults
= [MODIFIER]
= [SCOPE: CLASS]
=
=
= [IN]
=   deployer - Deployer
=       The deployment state.
=
=   synchronousDeploy - xsd:boolean
=       (optional) Specify whether to block until the changes have been
=       fully deployed to network. A value of "true" means to block.
=       A value of "false" means to return immediately.
=       Default: false (asynchronous)
=
=   clearOnDeployFailure - xsd:boolean
=       (optional) Specify whether clear the deployer if a failure occurs.
=       A value of "true" means to clear.
=       A value of "false" means to leave the failed deployer.
=       Default: false
=
=   deployRetries - xsd:int
=       (optional) The number of times to attempt re-deployment during
=       synchronous deployment. This parameter is meaningless in the
=       asynchronous case.
=       Default: 0
=
=   deployRetryInterval - xsd:long
=       (optional) The number of milliseconds to wait between deployment
=       retries. This parameter is meaningless in the asynchronous case.
=       Default: 0
=
=   baseInstanceFullName - xsd:string
=   alarmableInstanceFilter - FilterHolder
=   faultFilter - FilterHolder
=   scopeType - xsd:int
=   scopeDepth - xsd:int
=
= [OUT]
=

```

```
= [EXCEPTIONS]
= fm.FaultManager.clearFaultsException
-->
<xsd:element name="fm.FaultManager.clearFaults">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="deployer" type="Deployer" minOccurs="1"/>
      <xsd:element name="synchronousDeploy" type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="clearOnDeployFailure" type="xsd:boolean" minOccurs="0"/>
      <xsd:element name="deployRetries" type="xsd:int" minOccurs="0"/>
      <xsd:element name="deployRetryInterval" type="xsd:long" minOccurs="0"/>
      <xsd:element name="baseInstanceFullName" type="xsd:string" minOccurs="1"/>
      <xsd:element name="alarmableInstanceFilter" type="FilterHolder"
minOccurs="1"/>
      <xsd:element name="faultFilter" type="FilterHolder" minOccurs="1"/>
      <xsd:element name="scopeType" type="xsd:int" minOccurs="1"/>
      <xsd:element name="scopeDepth" type="xsd:int" minOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
<xsd:element name="fm.FaultManager.clearFaultsResponse">
  <xsd:complexType>
    <xsd:all>
      </xsd:all>
    </xsd:complexType>
  </xsd:element>
```


9 — *XML message structure*

- 9.1 XML message structure overview 9-2
- 9.2 CLI commands within XML methods 9-17
- 9.3 Mapping XML methods to GUI operations 9-18

9.1 XML message structure overview

The 5620 SAM-O uses request, response, and fault messages to handle the XML message data related to network objects. Messages are sent to the 5620 SAM-O using an RPC pattern. The messages are constructed and wrapped in a SOAP envelope. The 5620 SAM-O then returns a success response message or a failure fault message.

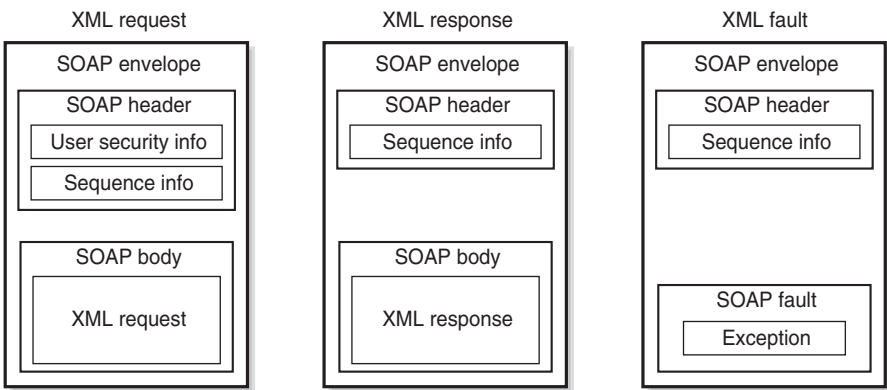
The SOAP encoding transports the XML data. Three types of SOAP messages are used:

- requests
- responses for a successful or unsuccessful execution
- faults for SOAP messages that are not well-formed

Message structure

Figure 9-1 shows the structure of the three SOAP message types.

Figure 9-1 SOAP message types



17289

Table 9-1 describes the SOAP message types.

Table 9-1 SOAP message type details

SOAP message type	Contents	Details
Request	SOAP envelope	Contains all message data
	SOAP header	<p>The header contains:</p> <ul style="list-style-type: none"> • user security details: <ul style="list-style-type: none"> • user IDs in plain text format • passwords in plain text or an MD5-hashed format. A utility to convert plain text passwords to an MD5-hashed format is available. See section 3.4 for more information. • sequence information in the form of a requestID to associate requests with responses and faults. You can use a string for the requestID. The requestID must be unique per HTTP request. <p>Do not use the following formats:</p> <ul style="list-style-type: none"> • packageName-packageName.ObjectName-[instanceFullNamelfApplicable].methodName. For example: netw-netw.Topology-[].configure • AreqObjectNameMethodName-CLIENT-userId-sequenceNumber For example: AreqVirtualInterfaceConfigureIpAddresses-CLIENT-admin-2 • SreqObjectNameMethodName-CLIENT-userId-sequenceNumber For example: SreqTopologyConfigure-CLIENT-operator-12
	SOAP body	<p>Contains the XML request in the general format:</p> <pre><package.object.method> or <internal_method> <inParam1>value</inParam1> <inParamX>value</inParamX> </package.object.method> or </internal_method></pre>
Response	SOAP envelope	Contains all message data
	SOAP header	Contains sequence information in the form of the request ID of the original request
	SOAP body	<p>Contains the XML response in the general format:</p> <pre><package.object.methodResponse> or <package.object.methodException> or <XMLException> or <IMException> <outParam1>value</outParam1> <outParamX>value</outParamX></pre>
Fault	SOAP envelope	Contains all message data
	SOAP header	Contains sequence information in the form of the request ID of the original request
	SOAP fault	<p>Contains:</p> <ul style="list-style-type: none"> • SOAP fault information • exception details <p>See “Faults and exceptions” for more information about the fault details.</p>



Note – The 5620 SAM-O supports streaming SOAP messages.

Requests

The 5620 SAM-O supports the following request types:

- synchronous
- asynchronous (default)

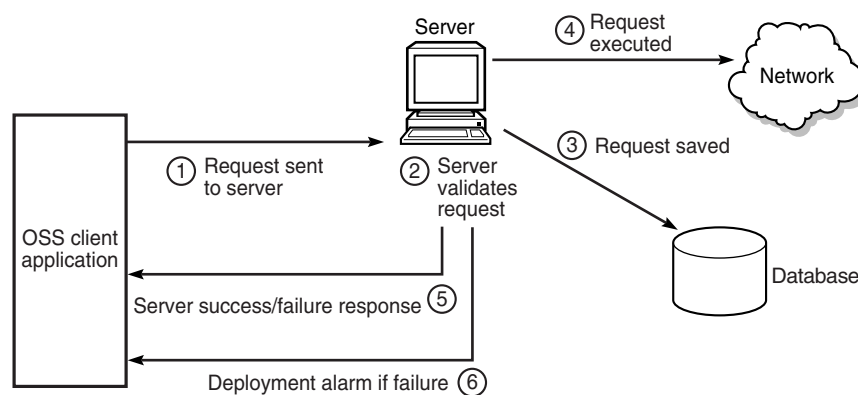
There are three scenarios for a message request:

- The OSS requests a database interaction that causes network deployments, for example, provisioning a service. See Figures 9-2 and 9-3.
- The OSS requests a database interaction that does not cause network deployments. See Figure 9-4.
- The OSS requests read information, for example, inventory information or an alarm feed. See Figure 9-5.

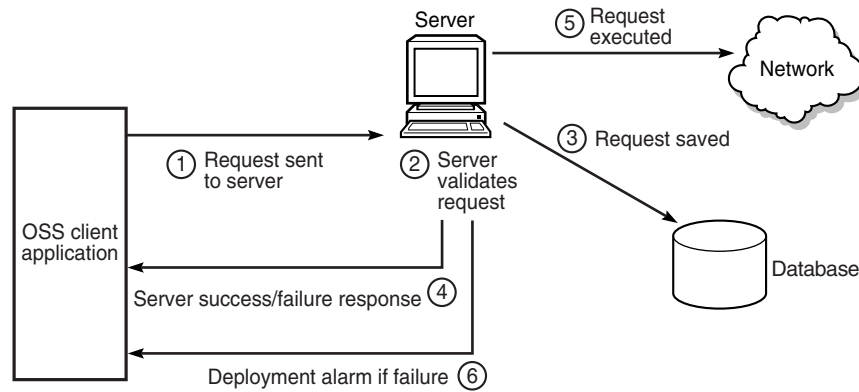


Note – The 5620 SAM maps the object FDNs to a corresponding database index. Alcatel-Lucent recommends using FDN properties in queries to maximize the processing speed of the query. The 5620 SAM-O queries should also use concrete classes rather than an abstract class to optimize performance.

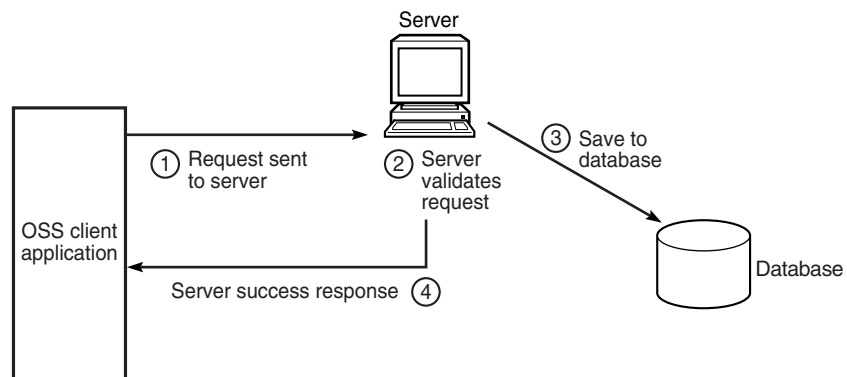
Figure 9-2 Database interaction with synchronous network deployment



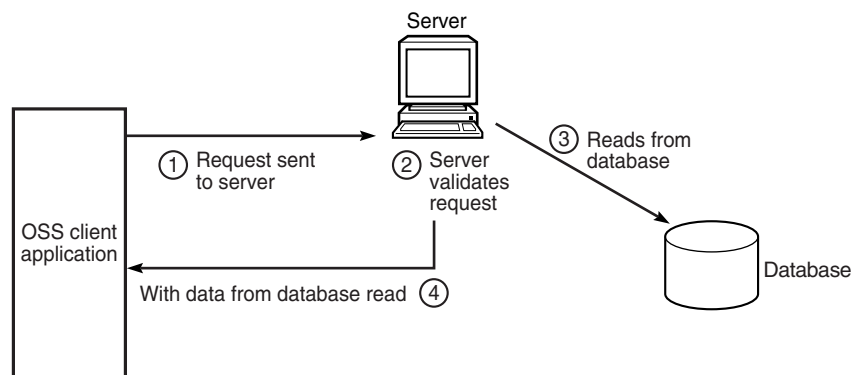
17752

Figure 9-3 Database interaction with asynchronous network deployment

17288

Figure 9-4 Database interaction without network deployment

17328

Figure 9-5 Read from database interaction

17287

Network interactions are performed using deployers. When a request fails before being committed to the database, a fault message is sent to the OSS client. For a synchronous request, the XML response indicates the success or failure of deployments. A single request may result in multiple deployers. The success or failure of multiple deployers is returned. OSS clients must be designed to accept more than one deployer in a response.

See “[Faults and exceptions](#)” in this section for more information about deployment failure recovery. See section [14.3](#) for more information about deployer failure JMS alarms and deployer request configurations.

Sample request

The XML/SOAP sample requests in this guide are intended as a base on which to build your own requests. The sample requests may contain:

- Methods that are deprecated by the 5620 SAM-O OSS
- Configuration information that is not applicable to your network architecture

Ensure that you test your request before network deployment.

Code [9-1](#) shows a sample request. The request shows the sequence of events and interactions associated with a request. In this sample, the OSS application changes the configuration of port 1/1/3 on node 10.1.202.93 to be access, dot1q, and administratively up.



Caution — The following sample message is an example of the request format. Use the sample as a base to build your request. Ensure that you test your request before network deployment.

Code 9-1: Sample request

```
<SOAP:Body>
  <generic.GenericObject.configureInstanceWithResult
xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <distinguishedName>network:10.1.202.93:shelf-1:cardSlot-2:card:daughterCards
lot-1:daughterCard:port-3</distinguishedName>
    <includeChildren>true</includeChildren>
    <configInfo>
      <equipment.PhysicalPort>
        <actionMask><bit>modify</bit></actionMask>
        <administrativeState>portInService</administrativeState>
        <mode>access</mode>
        <encapType>qEncap</encapType>
        <children-Set/>
      </equipment.PhysicalPort>
    </configInfo>
  </generic.GenericObject.configureInstanceWithResult>
</SOAP:Body>
```

Code [9-1](#) lists the following details for the request:

- The generic.GenericObject.configureInstanceWithResult identifies:
 - the use of the generic package
 - the executed method configureInstanceWithResult is defined in the GenericObject class
- The standards-based schema and version.

- The distinguishedName tag defines the unique Distinguished Full Name of the object in 5620 SAM.
- The configInfo object is a generic object that can contain various objects such as equipment.PhysicalPort

The configureInstanceWithResult method, as indicated in the request, is defined in the genericMethods.xsd. The equipmentTypes.xsd schema file defines the valid parameters available for configuration on equipment.PhysicalPort. The request is executed by the server according to the parameters specified in the configureInstanceWithResult method.

XML request grouping

You can concatenate multiple message requests in the body of a single SOAP message. The response contains the result of each request.

Code 9-2 shows an example of a SOAP message with multiple message requests.

Code 9-2: XML request grouping

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <security>
        <user>user name</user>
        <password>MD5-hashed password</password>
      </security>
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <find xmlns="xmlapi_1.0">
      <fullClassName>equipment.DaughterCard</fullClassName>
      <filter>
        <equal name="siteName" value="sim202_93"/>
      </filter>
      <resultFilter>
        <attribute>objectFullName</attribute>
        <attribute>operationalState</attribute>
        <attribute>administrativeState</attribute>
        <attribute>specificType</attribute>
      <children></children>
    </resultFilter>
  </find>
    <find xmlns="xmlapi_1.0">
      <fullClassName>equipment.PhysicalPort</fullClassName>
      <filter>
        <equal name="siteName" value="sim202_93"/>
      </filter>
      <resultFilter>
        <attribute>objectFullName</attribute>
        <attribute>operationalState</attribute>
        <attribute>administrativeState</attribute>
        <attribute>mode</attribute>
      <children></children>
    </resultFilter>
  </find>
</SOAP:Body>
```

Password format

The 5620 SAM server accepts XML requests with plaintext or MD5-hashed passwords. See section 3.4 for more information.

Action on failure

The `<continueOnFailure>` and `<onFailure>` options help manage failed requests.

```
<continueOnFailure>false</continueOnFailure>
```

When the `<continueOnFailure>` option is set to false and a request in a concatenated message fails, the 5620 SAM responds as follows:

- ignores all remaining requests in the concatenated message
- returns responses for all successful requests up to the failed request
- generates an exception message for the failure

If the `<continueOnFailure>` option is set to true, the execution of the requests continues even if a failure has occurred. Responses and exceptions are returned for each success or failure, respectively.

The `<execute>` and `<onFailure>` options allow requests to recover from a failure. Multiple message requests can be placed within `<execute>` and `<onFailure>` blocks. The execution begins with the first request in the `<execute>` block. If there is a failure, the execution branches to the start of the `<onFailure>` block. Returned results contain the results of all the requests up until the failed request in the `<execute>` block, and the results of all the requests up until the failed request, if any, in the `<onFailure>` block. Only one level of nesting is allowed inside an `<onFailure>` block.

Each `<execute>` block can have a timeout parameter. For example,

```
<execute timeout="500000" xmlns="xmlapi_1.0">
...
</execute>
<onFailure>
...
</onFailure>
```

Elapsed time is verified between requests in the `<execute>` block. If the timeout is exceeded, the execution is terminated and a request timed out failure is returned in the result.



Note — It is more difficult to troubleshoot problems with multiple message requests in comparison to single message requests. When failures occur, a corrective configuration action may need to be taken to address certain issues, such as a partial configuration.

Alcatel-Lucent recommends that you consider the risks associated with support and maintenance if you choose to use multiple message requests.

Code 9-3 shows a request to create a VPLS site.

Code 9-3: Sample request to create a VPLS site

```

<soapenv:Body>
  <generic.GenericObject.configureChildInstance
xmlns="xmlapi_1.0">
    <distinguishedName>svc-mgr:service-67</distinguishedName>
    <childConfigInfo>
      <vpls.Site>
        <actionMask>
          <bit>create</bit>
          <bit>modify</bit>
        </actionMask>
        <siteId>10.1.241.69</siteId>
        <displayName>Alcatel_Lab:VPLS:100020:sim241_2</displayName>
        <administrativeState>1</administrativeState>
        <mtu>0</mtu>
      </vpls.Site>
    </childConfigInfo>
  </generic.GenericObject.configureChildInstance>
</soapenv:Body>

```

Code 9-4 shows a request that can be used after a failure occurs when creating the VPLS site.

Code 9-4: Sample request after a failure during VPLS site creation

```

<soapenv:Body>
  <generic.GenericObject.deleteInstance xmlns="xmlapi_1.0">
    <distinguishedName>
      svc-mgr:service-67:10.1.241.69
    </distinguishedName>
  </generic.GenericObject.deleteInstance>
</soapenv:Body>

```

You can combine both requests in codes 9-3 and 9-4 with the execute and onFailure blocks, as shown in code 9-5.

Code 9-5: Sample request with execute and onFailure blocks

```

<soapenv:Body>
  <execute>
    <generic.GenericObject.configureChildInstance
xmlns="xmlapi_1.0">
      <distinguishedName>svc-mgr:service-67</distinguishedName>
      <childConfigInfo>
        <vpls.Site>
          <actionMask>
            <bit>create</bit>
            <bit>modify</bit>
          </actionMask>
          <siteId>10.1.241.69</siteId>
          <displayName>Alcatel_Lab:VPLS:100020:sim241_2</displayName>
          <administrativeState>1</administrativeState>
          <mtu>0</mtu>
        </vpls.Site>
      </childConfigInfo>
    </generic.GenericObject.configureChildInstance>
  </execute>

```

```
<onFailure>
<generic.GenericObject.deleteInstance xmlns="xmlapi_1.0">
  <distinguishedName>
svc-mgr:service-67:10.1.241.69
  </distinguishedName>
</generic.GenericObject.deleteInstance>
</onFailure>
</soapenv:Body>
```

Responses

Responses identify the execution of a method.



Note — Alcatel-Lucent recommends that you construct requests for one object at a time. For example, to configure individual cards or interfaces.

Before you create services, configure the related objects. Each step to create the service should be sent as an individual request rather than as a complex query. Smaller requests can simplify the debugging of 5620 SAM-O application interactions.

Code 9-6 shows a successful response to the request to configure a network interface.

Code 9-6: Sample response to a successful request to configure a network interface

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <rtr.RoutingInstanceSite.configureResponse xmlns="xmlapi_1.0">

<objectFullName>network:10.1.202.95:router-1:ip-interface-24</objectFullName>
    </rtr.RoutingInstanceSite.configureResponse>
  </SOAP:Body>
</SOAP:Envelope>
```

You can also include the content type information in the HTTP response header, using the `<xmlapi>` element in the `nms-server.xml` file. The `xmlContentType` attribute defines the content; for example, using the default attribute value of `text/xml; charset=ISO-8859-1`. You can use any valid string to configure the attribute.

Code 9-7 shows a response that includes the content type.

Code 9-7: Sample response that includes content type

```
HTTP/1.1 200 OK
  Request Version: HTTP/1.1
  Response Code: 200
Server: Apache-Coyote/1.1
X-Powered-By: Servlet 2.4; JBoss-4.0.2 (build: CVSTag=JBoss_4_0_2
date=200505022023)/Tomcat-5.5
Content-Type: text/xml; charset=ISO-8859-1
Transfer-Encoding: chunked
Date: Fri, 12 May 2006 16:55:02 GMT
```

HTTP response codes

Table 9-2 defines the HTTP response codes based on the RFC 2616 standard. The 5620 SAM-O does not use all responses defined by the RFC 2616 standard. The HTTP response codes are associated with the successful or unsuccessful transmission of the XML response.

Table 9-2 HTTP response codes

Code Range	Definition
1xx ⁽¹⁾	Informational - Request received, continuing process.
2xx	Success - The action was successfully received, understood, and accepted.
3xx ⁽¹⁾	Redirection - Further action must be taken in order to complete the request.
4xx	Client Error - The request contains bad syntax or cannot be fulfilled.
5xx	Server Error - The server failed to fulfill an apparently valid request.

Note

⁽¹⁾ The XML API does not send 1xx and 3xx messages.

The XML API client must check the HTTP response code before it attempts to parse the XML API results. An HTTP response code between 200 and 299 identifies successful execution of the request. The HTTP body for a 2xx response contains a valid XML SOAP response. All other response codes may not contain well-formed XML.

Faults and exceptions

The 5620 SAM-O produces a SOAP fault message when there is a problem processing a SOAP header or empty SOAP body. The subsequent errors result in an exception within the SOAP body.

SOAP exception messages can contain the following fault-related details:

- <faultcode> provides some standard SOAP information, including:
 - version mismatch between SOAP versions
 - server problems
 - client problems, for example, sending a request to execute a method that does not exist on the server
- <faultstring> error message
- <faultactor> is the URL of the server if the server is the problem

The 5620 SAM-O uses the following format for the <faultstring> element:

```
<faultstring>[error_string] message</faultstring>
```

Table 9-3 lists the error types that are associated with the <faultstring> element.

Table 9-3 <faultstring> error types for SOAP fault messages

[error_string]	Details
license	Not licensed to use the XML API
soap	Incomplete or incorrect SOAP construction
xml-header	XML API header is missing, or is missing necessary information, such as the user ID or the MD5-hashed password
security	Authentication error, for example, invalid user ID

Table 9-4 describes the faults and exceptions that can appear in an XML response.

Table 9-4 Summary of XML response faults and exceptions

Fault or exception	Format	Description
SOAPFault	<pre><SOAP:Fault> <faultcode>value</faultcode> <faultstring>value</faultstring> <faultactor>value</faultactor><detail> <requestID>value</requestID></detail> </SOAP:Fault></pre>	<p>Errors in a SOAP header.</p> <p>The <faultstring> in a <SOAP:Fault> response provides details on the fault.</p> <p>The strings may change over different releases of the 5620 SAM and therefore should not be parsed by the OSS application.</p>
XMLException	<pre><XMLException xmlns="xmlapi_1.0"> <description>value</description> <line>value</line> <column>value</column> </XMLException></pre>	<p>Errors in the XML syntax or because of a non-existent method or attribute in the 5620 SAM-O.</p> <p>The XMLException <description> field provides details on the XML errors.</p> <p>The strings may change over different releases of the 5620 SAM and therefore should not be parsed by the OSS application.</p>
IMException	<pre><IMException> <cause>value</cause> <description>value</description> </IMException></pre>	<p>Errors in the object model. For example, binding a SAP to a non-existent tunnel.</p> <p>The <description> field provides details about the error.</p> <p>The strings may change over different releases of the 5620 SAM and therefore should not be parsed by the OSS application.</p>
BaseException	<pre><BaseException xmlns="xmlapi_1.0"> <description>value</description> </BaseException></pre>	<p>Operation-related exceptions, such as configurationException and creationException. The BaseException appears in the XML response in these exceptions.</p> <p>The <description> field in the XML response provides details about the error.</p> <p>The strings may change over different releases of the 5620 SAM and therefore should not be parsed by the OSS application.</p>

Table 9-5 lists sample faults and exceptions that can appear in an XML response, as described in table 9-4.

Table 9-5 Sample faults and exceptions

Fault or exception	Example fault or exception
SOAPFault	<pre> <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP:Header> <header xmlns="xmlapi_1.0"> <requestID>XmlApiClient:0</requestID> </header> </SOAP:Header> <SOAP:Fault> <faultcode>SOAP:Client</faultcode> <faultstring>[security] Users require OSS Management privileges to use SAM-O</faultstring> <faultactor>XmlApi</faultactor> <detail> <requestID>XmlApiClient:0</requestID> </detail> </SOAP:Fault> </SOAP:Envelope> </pre>
XMLException	<pre> <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP:Header> <header xmlns="xmlapi_1.0"> <requestID>XmlApiClient:0</requestID> </header> </SOAP:Header> <SOAP:Body> <XMLException xmlns="xmlapi_1.0"> <description>Command 'fm.FaultManager.findFault2s' is not defined</description> <line>4</line> <column>57</column> </XMLException> </SOAP:Body> </SOAP:Envelope> </pre>
IMException	<pre> <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP:Header> <header xmlns="xmlapi_1.0"> <requestID>client1:0</requestID> </header> </SOAP:Header> <SOAP:Body> <IMException xmlns="xmlapi_1.0"> <cause>java.lang.SecurityException</cause> <description>SecurityManager:Security Breach: No such user: 'oss_client'</description> </IMException> </SOAP:Body> </pre>

(1 of 2)

Fault or exception	Example fault or exception
BaseException	<pre> <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP:Header> <header xmlns="xmlapi_1.0"> <requestID>XmlApiClient:5</requestID> </header> </SOAP:Header> <SOAP:Body> <script.ScriptManager.configureException xmlns="xmlapi_1.0"> <description>[app: script] [class: script.Script] [instance: -unknown-] [descr: the NE type is invalide type,{neType=craig}] </description> </script.ScriptManager.configureException> </SOAP:Body> </SOAP:Envelope> </pre>

(2 of 2)

Code 9-8 shows an exception message generated when a user does not have the required OSS management privileges to use SAM-O. The sample exception provides the following details:

- <faultcode> element indicates that the server could not execute the request because of a fault of the client
- <faultstring> element provides the security error string, which indicates that the Client user specified does not have the required OSS management privileges to use SAM-O
- <faultactor> element identifies the XmlApi as the fault factor

Code 9-8: Sample exception message for failed router configuration

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <requestID>clientName@requestId</requestID>
      <requestTime>Jan 2, 2007 1:18:53 PM</requestTime>
      <responseTime>Jan 2, 2007 1:18:53 PM</responseTime>
    </header>
  </SOAP:Header>
  <SOAP:Fault>
    <faultcode>SOAP:Client</faultcode>
    <faultstring>[security] Users require OSS Management privileges to use
SAM-O</faultstring>
    <faultactor>XmlApi</faultactor>
    <detail>
      <requestID>clientName@requestId</requestID>
    </detail>
  </SOAP:Fault>
</SOAP:Envelope>

```



Note — The <responseTime> tag in a 5620 SAM-O header is the time at which the response stream is opened.

The <detail> element in the SOAP exception message identifies the command that caused the request failure.

The 5620 SAM-O validates the XML request when it is sent to the server. If the validation fails, 5620 SAM-O generates and sends an exception message as shown in Code 9-8. If the request requires changes to the managed network, deployers are created and queued to send the changes to the routers. If the deployment fails, 5620 SAM-O raises an alarm which includes details for the particular deployer ID. The network can be in an indeterminate state because another configuration request may have succeeded. See section 14.3 for more information about creating and viewing deployer requests and alarms.

To recover from request errors, the OSS application may need to check the affected objects in the deployer and issue appropriate requests to undo the action that caused the error. Another option is to manually maintain a list of failures for resolution at a later date.



Note — See “[Workflow to handle deployer failures](#)” in chapter 14 for more information about how to handle deployment errors that may cause the 5620 SAM database to be out of synchronization with the managed network.

Logging exception messages

Additional troubleshooting information about exceptions is available in the server EmsServer.log file on the 5620 SAM server. To obtain the log file, go to the 5620 SAM server installation directory and navigate to the /nms/log/server directory. You can then open the log file using a text editing tool. You can view the backup files in the same directory as the original file. See section 9.3 for more information.

Log file for XML requests, responses, and exceptions

You can enable the logging of 5620 SAM-O XML request, response, and exception messages on the active or standby 5620 SAM servers. The entries in the message logs can assist you in the following tasks:

- identifying the user associated with an XML request
- debugging the OSS requests from a user

You can enable XML message logging for an individual user or multiple users. See Procedure A-6 for more information. The 5620 SAM creates log files for each HTTP request and response associated with the 5620 SAM-O user defined by the <systemOssiLog> element in the nms-server.xml file. An HTTP request and response can contain multiple XML requests and responses.

The 5620 SAM stores the log file in the directory defined by the <systemOssiLog> element in the nms-server.xml file, typically ../log/. The 5620 SAM uses the following naming convention for log files:

- ossi<user>Request<n+>.log
 - ossi<user>Response<n+>.log
- where <user> is the 5620 SAM user name and <n+> is the incremental request number

The request log contains the body of the SOAP message. The response log contains the entire SOAP envelope of the response.



Caution — The prolonged use of XML message logging in a busy network environment may have network management performance impacts. Use the log functionality for a limited time to debug a specific problem.

Code 9-9 shows a sample request for an XML message log file.

Code 9-9: Sample request from the ossiuserRequest1.log XML message log file

```
<?xml version='1.0'?>
<Request user="user" requestId="clientName@requestId">
  <find>
    <fullClassName>...</fullClassName>
    <filter>
      ...
    </filter>
  </find>
</Request>
```

Code 9-10 shows sample response to the request for an XML message log file.

Code 9-10: Sample response from the ossiuserResponse1.log XML message log file

```
<?xml version='1.0'?>
<Response user="user" requestId="clientName@requestId">
  <SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Header>
      <header xmlns="xmlapi_1.0">
        <requestID>clientName@requestId</requestID>
        <requestTime>Jan 2, 2007 3:10:41 PM</requestTime>
        <responseTime>Jan 2, 2007 3:10:41 PM</responseTime>
      </header>
    </SOAP:Header>
    <SOAP:Body>
      <findResponse xmlns="xmlapi_1.0">
        <result>
          ...
        </result>
      </findResponse>
    </SOAP:Body>
  </SOAP:Envelope>
</Response>
```



Note — The <responseTime> tag in a 5620 SAM-O header is the time at which the response stream is opened.

The system administrator configures the XML message log using the nms-server.xml file in the install directory. The system administrator is responsible for the maintenance and backup of log files.



Caution — Modifying the nms-server.xml file can seriously affect the network management and performance of the 5620 SAM.

As part of debug logging policy configuration, the system administrator can configure the maximum disk space for storing log messages. The 5620 SAM disables the log option and raises an alarm if the size of the log file exceeds the storage allocation. The default log file retention period is 24 hours. See Procedure A-6 for information about how to modify the log file retention period.

In a redundant network configuration, the 5620 SAM logs the activities for the active server.

See Procedure A-6 for information about how to enable the logging of XML message transactions on the 5620 SAM server. You can also use the 5620 SAM GUI client to view activity for a 5620 SAM-O user. The 5620 SAM GUI client tracks the network management activity for GUI and OSS users. See the *5620 SAM User Guide* for more information.

9.2 CLI commands within XML methods

The 5620 SAM client can send CLI commands to a node using methods in the network element object. The client receives the corresponding response for the executed CLI commands. Table 9-6 describes the methods in the network element object.

Table 9-6 CLI methods in the network element object

Method	Description
executeCli	Sends a CLI command to the node. See Code 9-11 for an XML example.
executeMultiCli	Sends multiple CLI commands to the node. See Code 9-12 for an XML example.

Code 9-11: Sample single CLI command

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

  <SOAP:Header>

    <header xmlns="xmlapi_1.0">
      <security>
        <user>user</user>
        <password>MD5-hashed user password</password>
      </security>
      <requestID>clientName@requestId</requestID>    </header>
    </SOAP:Header>
    <SOAP:Body>
      <netw.NetworkElement.executeCli
        xmlns="xmlapi_1.0">

        <instanceFullName>network:10.1.186.183</instanceFullName>
        <command>ping 138.120.186.185</command>
      </netw.NetworkElement.executeCli>
    </SOAP:Body>
  </SOAP:Envelope>
```

Code 9-12: Sample multiple CLI commands

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

```
<SOAP:Header>

  <header xmlns="xmlapi_1.0">
    <security>
      <user>user</user>
      <password>MD5-hashed user password</password>
    </security>
    <requestID>clientName@requestId</requestID>          </header>
  </SOAP:Header>
  <SOAP:Body>
    <netw.NetworkElement.executeMultiCli
      xmlns="xmlapi_1.0">

      <instanceFullName>network:10.1.186.183</instanceFullName>

      <commands>
        <string>ping 138.120.186.185</string>
        <string>ping 138.120.186.184</string>
      </commands>
    </netw.NetworkElement.executeMultiCli>
  </SOAP:Body>
</SOAP:Envelope>
```

9.3 Mapping XML methods to GUI operations

In development environments, you can configure the server to record the methods related to 5620 SAM GUI operations in the server log. The additional log information helps:

- developers understand which methods should be used to integrate an OSS application. You can use the server log file to trace the classes and methods called when an action is performed on the GUI.
- Alcatel-Lucent support staff troubleshoot the operation of the 5620 SAM

Procedure 9-1 To enable logging of GUI operations on the 5620 SAM server



Caution — The logging of GUI operations should only be enabled on servers in a development environment. When you enable the server log on a server that is running in live network environment, server performance may be affected.

- 1 Open the `nms-server.xml` file on the 5620 SAM server. The file is located in the *install directory*/`nms/config` directory, where *install directory* is the installation directory.
- 2 Find the `<Filter>` element within the `<systemLog>` element.
- 3 Add the following code directly after the line that starts with `<!-- Don't change this section END -->`.

```
<include severity="debug" class=".*IFGBase"
method="printDebug.*"/>
```



Caution — Do not modify other `nms-server.xml` parameters. Modifying the file can seriously affect the network management and performance of the 5620 SAM.

- 4 Restart the 5620 SAM server. All actions executed on the GUI and the XML OSS interface are now logged to the server log file.
 - 5 Open the EmsServer.log debug log file. The log file is located in the *install directory/nms/log/server* directory, where *install directory* is the installation directory.
-

Procedure 9-2 To view the GUI operation in the server log

- 1 Complete Procedure 9-1.
 - 2 Open the server log file.
 - 3 Perform an action on the GUI.
 - 4 View the logged information for that action in the server log file to determine the class and method used.
-

Output format of a GUI operation in the server log file

Code 9-13 shows the output format of a GUI operation in the server log file after you delete an IP interface on the GUI.

Code 9-13: Sample server log output

```
<server.core.IFGBase.printDebug>
IFG[generic]:([generic.GenericObject].deleteInstance() : BEGIN
<server.core.NullDeployer.issueDatabaseLog> From RIWorker
[8](1301414107317),DatabaseLogRuleInvokationCapsule User Id:
securityManager:user-admin Request Id:
AreqGenericObjectDeleteInstance-CLIENT-admin-5620SAM@6-181 Target Class:
rtr.DirectInterfaceCtp Target Object Id:
network:35.121.20.59:router-1:ip-interface-2:TP Operation: ObjectDeletion
Result: true
<server.core.IFGBase.printDebug>
IFG[generic]:([generic.GenericObject].deleteInstance() : END ::: successfully
executed : true
```

Code 9-13 contains the following information:

- the class and method that corresponds to the GUI action, which in this example is `generic.GenericObject.deleteInstance`. You can use the identified class and method to create an OSS application that performs the same task.
- the specific instance of the action that was on the managed router IP interface with the unique ID `network:35.121.20.59:router-1:ip-interface-2:TP`. The unique ID identifies that the router has an IP address of 35.121.20.59 and a name of router-1.
- acknowledgement of successful completion of the configuration action



Note 1 – In some cases, a GUI action may use custom methods that are not directly applicable for use by an OSS. For example, when configuring network objects using the GUI, the server log output could log the `mpls.LspPath.configure` method. In this case an OSS should use the generic method `generic.GenericObject.configureInstance`. See section 14.2 for more information about using generic methods for configuring network objects.

Note 2 – In some cases, no specific instance is returned, for example, when you perform a find action.

OSS domains

- 10 – Fault management
- 11 – OAM
- 12 – Inventory management
- 13 – Accounting and performance monitoring

10 – Fault management

- 10.1 Fault management overview 10-2
- 10.2 Communicating with the 5620 SAM 10-2
- 10.3 Alarm sources 10-2
- 10.4 Alarm definitions 10-3
- 10.5 Alarm messages 10-3
- 10.6 Retrieving alarms 10-6
- 10.7 Alarm management 10-6
- 10.8 Alarm policies 10-7
- 10.9 Alarm correlation 10-7
- 10.10 OSS client alarm testing 10-7
- 10.11 Workflow to set up alarm management 10-9

10.1 Fault management overview

The OSS applications that focus on fault management use events and alarms to detect, log, notify users of, and automatically correct network problems to keep the network running effectively.

10.2 Communicating with the 5620 SAM

An OSS fault management application can use a JMS and/or an XML/SOAP interface to communicate with the 5620 SAM server. For alarm management:

- The 5620 SAM sends network event/alarm notifications using the JMS interface in near-real-time to the OSS. An OSS uses the JMS event stream to monitor new alarms and status changes on existing alarms (such as alarm clearing or escalation/de-escalation). See chapter 4 for more information about connecting to the JMS event stream.
- The OSS uses XML/SOAP to send requests for retrieving information about objects with alarms, historical alarm information, and configuring alarms. See chapter 5 for more information about the XML API.

The following sections provide information about JMS messages and API calls that are relevant to a fault management OSS developer.



Note — Alcatel-Lucent recommends that a fault management OSS use the JMS event stream for ongoing monitoring of alarms and network objects, and the XML API only for on-demand information retrieval at system startup or when a resynchronization is required.

10.3 Alarm sources

The sources that initiate 5620 SAM alarm notifications are:

- network traps
- 5620 SAM alarms

Network traps

The 5620 SAM monitors network health by listening for traps from the managed network elements. These SNMP traps may indicate conditions that include configuration or operational state changes, security breach attempts, and equipment faults.

When the 5620 SAM detects a fault in the network, one or more alarms are generated, depending on the cause of the fault. When an alarm is created, modified or deleted the JMS messages that contain the alarm information are sent to interested listeners.



Note — Although the 5620 SAM receives network information in traps and generating alarms about fault conditions in the network, the traps are not directly mapped to alarms. For example, one trap may generate more than one JMS message.

5620 SAM alarms

The 5620 SAM generates alarms based on network conditions; the alarms do not apply to specific NEs. The alarms provide information including: faults and changes to the 5620 SAM server system, database, redundancy, users, and logs.

10.4 Alarm definitions

An alarm in the 5620 SAM is represented by an instance of `fm.AlarmObject`. The attributes of the `AlarmObject` are described in the *5620 SAM-O XML Reference*, and in section 10.5.

Information about the alarm types are also available. See the *5620 SAM Alarm Reference* for more information.

When an OSS receives an alarm from the 5620 SAM-O, the OSS can map to the corresponding `alarmDetails` record using the fields described in Table 10-1.

Table 10-1 Alarm mapping details

fm.AlarmObject attribute	Corresponding alarmDetails field	Comments
alarmClassTag	alarmClassTag	alarmClassTag is unique in the alarmDetails list
alarmName	alarmName	The combination of alarmName and alarmType is unique in the alarmDetails list
type	alarmType	

10.5 Alarm messages

An OSS needs to subscribe to JMS in order to receive the notification of new alarms or changes to existing alarms; for example, escalation/de-escalation, acknowledgement, or recurrence. See section 4.6 for information about properties that are common to all JMS events. See Table 4-5 for properties that are specific to fault messages.

When an alarm is generated, the 5620 SAM sends a JMS message with an `fm.AlarmInfo` structure. Code 10-1 shows a sample message for a new alarm. The fields of the `AlarmInfo` correspond to the properties of the new `fm.AlarmObject`. See the *5620 SAM-O XML Reference* for `fm.AlarmInfo` and `fm.AlarmObject` information.

Code 10-1: Sample message for a new alarm

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>ObjectCreation</eventName>
      <ALA_category>FAULT</ALA_category>
      <ALA_isCorr>false</ALA_isCorr>
      <ALA_isVessel>false</ALA_isVessel>
      <ALA_alarmType>equipmentAlarm</ALA_alarmType>
    </header>
  </SOAP:Header>
</SOAP:Envelope>
```

```

    <ALA_allomorphic>fm.AlarmObject</ALA_allomorphic>
    <MTOSI_NTType>NT_ALARM</MTOSI_NTType>
    <MTOSI_serviceAffecting>>false</MTOSI_serviceAffecting>
    <MTOSI_probableCause>inoperableEquipment</MTOSI_probableCause>
    <ALA_clientId/>
    <MTOSI_aliasNameList>EquipmentDown</MTOSI_aliasNameList>
    <ALA_OLC>2</ALA_OLC>
    <MTOSI_osTime>1315841370394</MTOSI_osTime>
    <MTOSI_objectName>faultManager:network@35.121.20.47@shelf-1@cardSlot-1@card@daughter
CardSlot-1@daughterCard@port-32|alarm-10-3-8</MTOSI_objectName>
    <MTOSI_perceivedSeverity>major</MTOSI_perceivedSeverity>
    <ALA_span>2:</ALA_span>
    <MTOSI_objectType>fm.AlarmObject</MTOSI_objectType>
    <ALA_eventName>ObjectCreation</ALA_eventName>
  </header>
</SOAP:Header>
<SOAP:Body>
  <jms xmlns="xmlapi_1.0">
    <objectCreationEvent>
      <fm.AlarmInfo>
        <severity>major</severity>
        <previousSeverity>indeterminate</previousSeverity>
        <originalSeverity>major</originalSeverity>
        <highestSeverity>major</highestSeverity>
        <probableCause>8</probableCause>
        <alarmName>10</alarmName>
        <type>3</type>
        <affectedObjectFullName>network:35.121.20.47:shelf-1:cardSlot-1:card:daughterC
ardSlot-1:daughterCard:port-32</affectedObjectFullName>
        <affectedObjectClassName>equipment.PhysicalPort</affectedObjectClassName>
        <isAcknowledged>>false</isAcknowledged>
        <wasAcknowledged>>false</wasAcknowledged>
        <acknowledgedBy>N/A</acknowledgedBy>
        <clearedBy>N/A</clearedBy>
        <deletedBy>N/A</deletedBy>
        <firstTimeDetected>1315841369120</firstTimeDetected>
        <lastTimeDetected>1315841369120</lastTimeDetected>
        <lastTimeSeverityChanged>0</lastTimeSeverityChanged>
        <lastTimeCleared>0</lastTimeCleared>
        <lastTimePromoted>0</lastTimePromoted>
        <lastTimeDemoted>0</lastTimeDemoted>
        <lastTimeEscalated>0</lastTimeEscalated>
        <lastTimeDeEscalated>0</lastTimeDeEscalated>
        <lastTimeAcknowledged>0</lastTimeAcknowledged>
        <frequency>0</frequency>
        <occurrences>N/A</occurrences>
        <numberOfOccurrences>1</numberOfOccurrences>
        <numberOfOccurrencesSinceClear>1</numberOfOccurrencesSinceClear>
        <numberOfOccurrencesSinceAck>0</numberOfOccurrencesSinceAck>
        <isServiceAffecting>>false</isServiceAffecting>
        <additionalText>N/A</additionalText>
        <operatorAssignedUrgency>indeterminate</operatorAssignedUrgency>
        <urgencyAssignedBy>N/A</urgencyAssignedBy>
        <relatedObjects>
          <null/>
        </relatedObjects>
        <affectingObjects>
          <null/>
        </affectingObjects>
        <subscriberId>0</subscriberId>
        <nodeId>35.121.20.47</nodeId>
        <nodeName>sim20_47</nodeName>
        <affectedObjectDisplayedName>Port 1/1/32</affectedObjectDisplayedName>
        <applicationDomain>Physical Equipment</applicationDomain>
        <displayedClass>PhysicalPort</displayedClass>
        <alarmClassTag>equipment.EquipmentDown</alarmClassTag>
        <affectedObjectClassIndex>87</affectedObjectClassIndex>
        <affectedObjectInstanceIndex>2018</affectedObjectInstanceIndex>
        <deployerName/>
        <deployerId>0</deployerId>
      </fm.AlarmInfo>
    </objectCreationEvent>
  </jms>
</SOAP:Body>
</SOAP:Envelope>

```

```

        <requestId>N/A</requestId>
        <requestUser>N/A</requestUser>
        <correlatingAlarm>N/A</correlatingAlarm>
        <isImplicitlyCleared>true</isImplicitlyCleared>
        <numberOfCorrelatedAlarms>0</numberOfCorrelatedAlarms>
        <olcState>inService</olcState>
        <objectFullName>faultManager:network@35.121.20.47@shelf-1@cardSlot-1@card@daug
hterCardSlot-1@daughterCard@port-32|alarm-10-3-8</objectFullName>
        <objectClassName>fm.AlarmObject</objectClassName>
        <allomorphicClassName>fm.AlarmObject</allomorphicClassName>
        <objectId>606777272</objectId>
        <displayName>Port 1/1/32 -
network@35.121.20.47@shelf-1@cardSlot-1@card@daughterCardSlot-1@daughterCard@port-32|alarm-10
-3-8</displayName>
        <lifeCycleState>1</lifeCycleState>
        <deploymentState>0</deploymentState>
        <neId>35.121.20.47</neId>
        <spanObjectPointer>node-35.121.20.47</spanObjectPointer>
        <parentClassName>fm.FaultManager</parentClassName>
        <unsetProperties/>
    </fm.AlarmInfo>
  </objectCreationEvent>
</jms>
</SOAP:Body>
</SOAP:Envelope>

```

Code 10-2 shows an attribute value change for a recurring alarm. Each time that the alarm recurs, the number of occurrences (both absolute and since the last clear) are incremented and the time last detected is updated. Code 10-2 shows the old and new values for each of the fields. See also “AttributeValueChangeEvent” in chapter 4.

Code 10-2: Sample attribute value change for a recurring alarm

```

<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <eventName>AttributeValueChange</eventName>
      <ALA_category>FAULT</ALA_category>
      <ALA_OLC>2</ALA_OLC>
      <ALA_isVessel>false</ALA_isVessel>
      <ALA_allomorphic>fm.AlarmObject</ALA_allomorphic>
      <MTOSI_osTime>1316464116127</MTOSI_osTime>
      <MTOSI_NTType>NT_ALARM</MTOSI_NTType>
      <MTOSI_objectName>faultManager:network@3.3.3.3@router-1@ospf-v2@areaSite-0.0.0.0@in
terface-18|alarm-40-11-43-packetSourceIpAddress=10.1.3.6</MTOSI_objectName>
      <ALA_span>:2:</ALA_span>
      <ALA_clientId/>
      <MTOSI_objectType>fm.AlarmObject</MTOSI_objectType>
      <ALA_eventName>AttributeValueChange</ALA_eventName>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <jms xmlns="xmlapi_1.0">
      <attributeValueChangeEvent>
        <objectFullName>faultManager:network@3.3.3.3@router-1@ospf-v2@areaSite-0.0.0.0@in
terface-18|alarm-40-11-43-packetSourceIpAddress=10.1.3.6</objectFullName>
        <attribute>
          <attributeName>numberOfOccurrencesSinceClear</attributeName>
          <newValue>
            <long>3490</long>
          </newValue>
          <oldValue>
            <long>3489</long>
          </oldValue>
        </attribute>
        <attribute>
          <attributeName>lastTimeDetected</attributeName>
          <newValue>

```

```
        <long>1316464116127</long>
      </newValue>
      <oldValue>
        <long>1316463991997</long>
      </oldValue>
    </attribute>
    <attribute>
      <attributeName>numberOfOccurences</attributeName>
      <newValue>
        <long>3490</long>
      </newValue>
      <oldValue>
        <long>3489</long>
      </oldValue>
    </attribute>
  </attributeValueChangeEvent>
</jms>
</SOAP:Body>
</SOAP:Envelope>
```



Note — The JMS subscriptions should include filters that are as restrictive as possible. See section 4.7 for more information about JMS message filtering.

10.6 Retrieving alarms

While ongoing monitoring of alarms is done using JMS, an OSS may need to retrieve alarms at system startup or after losing sync with the 5620 SAM. See section 4.9 for information about situations in which an OSS may lose synchronicity with the 5620 SAM.

The `fm.FaultManager` provides several convenience methods for retrieving faults. The methods should be used instead of `find` or `findToFile` for `fm.AlarmObjects`. The most common method for retrieving alarm information is `findFaults`. The method provides options to filter alarms and display a list of alarms. See the `fm.FaultManager` in the *5620 SAM-O XML Reference* for all of the methods available, and the faults directory in Appendix B for examples of retrieving faults.



Note — Requests should use filter the number of objects returned using `filter/faultFilter` where applicable to the `find` method used. Requests should also limit the attributes returned to attributes that are of specific interest, using the `resultFilter`. See section 12.3 for more information about filters. See `Fault/findFaultsWithFilter.xml` in Appendix B for an example of filtering alarms. There are also several examples of using filters and `resultFilters` in the basic code samples.

10.7 Alarm management

Most common operations on alarms, including promotion/demotion, acknowledgement, and clearing, can be performed using convenience methods that are provided by `fm.FaultManager`. These methods take either a single alarm or set of alarms as arguments. See the *5620 SAM-O XML Reference* for more information. For code samples, see the Fault directory in Appendix B.

10.8 Alarm policies

Alarm policies are used to control how the 5620 SAM handles incoming alarms. For example, an OSS can manipulate the global policy (`fm.GlobalPolicy`), which applies to all of the alarms, or specific policies that apply to each type of alarm (`fm.SpecificPolicy`). There is one global policy for the entire system, and one specific policy for each alarm type. Both types of policies are configured using `generic.GenericObject.configureInstanceWithResult`. See Code 9-1 and the *5620 SAM-O XML Reference* for more information about the methods. See Appendix B for a sample request to configure a specific alarm policy.

10.9 Alarm correlation

A correlated alarm indicates that a fault on an object is caused by a fault on another object. For example, if a port goes down, an alarm is generated for the port, and correlated alarms are generated for services that rely on the port. To ignore correlated alarms in the JMS stream, filter out alarms where `ALA_isCorr` is true.



Note — The `ALA_isCorr` is an optional property, and is not included in all of the events. See “[Optional properties](#)” in section 4.7 for information about how to filter optional properties.

See the alarm management chapter in the *5620 SAM User Guide* for more information about correlated alarms.

10.10 OSS client alarm testing

The following functions can be used for testing fault management OSS applications: test alarms, and the JMS record and playback tool.

Test alarms

You can use the `fm.FaultManager.testAlarm` method to initiate an alarm that is generated by the 5620 SAM and sent to the OSS client using JMS. The `fm.FaultManager.testAlarm` method allows you to generate alarms using the 5620 SAM and facilitates alarm testing using an OSS client. The `fm.FaultManager.testAlarm` method allows you to manually generate alarms instead of using equipment traps to initiate the alarm condition.



Note 1 — You must clear the alarms that are associated with the `fm.FaultManager.testAlarm` method after you complete the alarm testing session.

Note 2 — The `fm.FaultManager.testAlarm` method does not perform a validation to ensure that you are generating an alarm with the correct parameters for the correct object class. The generated alarm does not have any correlation, aggregation, or association relationships with other objects or alarms.

To create an alarm that simulates a real alarm, you must include the following alarm attributes when you use the `fm.FaultManager.testAlarm` method:

- `objectInstanceName`
- `alarmNameId`
- `alarmTypeId`
- `probableCauseId`
- `severity`
- `alarmClassTag`

See the *5620 SAM Troubleshooting Guide* for more information about alarm attributes.

The `fm.FaultManager.testAlarm` method can be used in a development environment to test specific types of alarms that require real NEs, where real NEs are not accessible. It is an effective means to generate alarms and verify the alarm response output that is received by an OSS client. Code 10-3 shows an example of how to generate a service site down alarm.

Code 10-3: Sample request to generate a service site down alarm

```
<SOAP:Body>
  <fm.FaultManager.testAlarm xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <objectInstanceName>svc-mgr:service-3:10.1.1.90</objectInstanceName>
    <alarmNameId>97</alarmNameId>
    <alarmTypeId>17</alarmTypeId>
    <probableCauseId>84</probableCauseId>
    <severity>critical</severity>
    <alarmClassTag>svc.ServiceSiteDown</alarmClassTag>
    <namingComponent/>
    <additionalText>my additional text</additionalText>
  </fm.FaultManager.testAlarm>
</SOAP:Body>
```

Code 10-4 shows an example of the response to the request to generate a service site down alarm.

Code 10-4: Sample response to the request to generate a service site down alarm

```
<SOAP:Body>
  <fm.FaultManager.testAlarmResponse xmlns="xmlapi_1.0"/>
</SOAP:Body>
```

JMS record and playback

The JMS record and playback tool can be used to collect JMS messages from a 5620 SAM network and review the messages in a test environment. See section 4.12 for more information about the JMS record and playback tool.

10.11 Workflow to set up alarm management

The following workflow lists the high-level steps required to set up alarm management.

- 1 Establish a JMS subscription. See section [4.11](#) for more information about creating a JMS consumer.
- 2 Retrieve inventory information for the required elements in the network. See Procedure [4-1](#) for how to perform a retrieval without losing events during a retrieval of inventory.
- 3 If required, retrieve objects that affect or are related to the required network objects using `fm.FaultManager.findObjectsAffectionOfn` and `fm.FaultManager.findObjectsRelatedToOfn` methods. See the *5620 SAM-O XML Reference* for more information about the methods.
- 4 Retrieve alarms for the network elements of interest using `fm.FaultManager.findAlarmsForOfn`. See the *5620 SAM-O XML Reference* for more information.
- 5 As required, configure alarm policies. See section [10.8](#) for more information.
- 6 As required, acknowledge, clear, or promote/demote the alarms. See section [10.7](#) for more information.

11 – OAM

11.1	OAM overview	11-2
11.2	Key packages and classes	11-2
11.3	Tests	11-2
11.4	Test suites	11-3
11.5	Generated tests	11-4
11.6	Workflow to generate tests	11-4
11.7	OAM Test Results file retrieval	11-4

11.1 OAM overview

The 5620 SAM allows you to create, configure, execute, schedule, and collect results for OAM tests, for network troubleshooting, and to verify compliance with SLAs.

This chapter covers topics of interest to an OAM OSS developer, with focus on the 5620 SAM-O interface. For more information about 5620 SAM OAM functionality, see the OAM and Service Test Manager chapters in the *5620 SAM User Guide*. For the code samples discussed in this chapter, as well as other OAM-related sample code, see [Appendix B](#).

11.2 Key packages and classes

The `sas` package defines common interfaces and operations for service assurance, OAM tests, and test policies and suites. The package also defines abstract test classes such as `sas.Ping` and `sas.Trace`, which inherit from `sas.Test`. Specific tests inherit from the abstract classes, and are defined in the package of the objects that they test. For example, LSP Ping and LSP Trace are subclasses of `sas.Ping` and `sas.Trace`, and are defined in the `mpls` package. See the main page of the *5620 SAM-O XML Reference* for a list of package descriptions.

11.3 Tests

The categories of service assurance tests are:

- ping tests (subclasses of `sas.Ping`)
- trace tests (subclasses of `sas.Trace`)
- actions (subclasses of `sas.Action`)

The types of each test are:

- test (subclass of `sas.Test`)
- deployed test (subclass of `sas.DeployedTest`)
- test definition (subclass of `sas.TestDefinition`)

Generally, deployed tests are transient objects that are created when the `sas.Test` is run. The deployed test is removed after the test is completed. The results are retrieved from the node and associated with the `sas.Test`.



Note — The 5620 SAM supports OmniSwitch ping and traceroute tests via user-defined CLI scripts instead of `sas` package objects. For sample scripts, see the sample OmniSwitch ping and traceroute CLI scripts in the *5620 SAM User Guide*. For information about deploying CLI scripts using the OSS, see [section 9.2](#).

To create a test using the OSS interface, use `generic.GenericObject.configureChildInstance`. The `CreateServiceTunnelPing.xml` file in the code samples demonstrates the creation of a `svt.TunnelPing` test with default values. See the *5620 SAM-O XML Reference* for the test class and additional test properties. See [Appendix B](#) for more information about the SDK code samples.

To run an existing test, use the `executeAndWait` method of `sas.Test`. See the *5620 SAM-O XML Reference* for method parameters. Several sample usages are available in the XML code samples; for example, `LspPing-UserExisting.xml`.

The `sas.Test` object provides a convenience method to create and run a test using an XML request. See `sas.Test.adhocExecuteAndWait` in the *5620 SAM-O XML Reference* for more information. Several sample usages are also in Appendix B; for example, `LspPing-CreateAndUse.xml`.

See the *5620 SAM User Guide* for a list of the OAM diagnostic tests that are supported by the 5620 SAM. Classes for most tests are in the packages for the tested elements; for example, ATM ping is defined in the `atm` package. The only exception to this is Ethernet CFM tests, which can be found in the `ethernetoam` package. See section 6.2 for information about mapping entity types to packages.

11.4 Test suites

Test suites are used to group OAM tests. After the suites are created, they can be scheduled to provide periodic results or run on demand to investigate service issues. Test in a suite are organized in the following groups:

- first-run tests
- generated tests
- last-run tests

First-run tests typically include high-level diagnostics, such as service-side or VPRN ping. The tests can be run sequentially or in parallel. Generated tests are run after the first-run tests. See section 11.5 for more information about generated tests. Last-run tests typically include transport-layer diagnostics, such as an LSP trace or a tunnel ping, and can also be run sequentially or in parallel.

Test suites are usually associated with a specific type of tested entity, such as VLL service or VPLS. You can also create a test suite with a mix of tested entities by setting `testedEntityType` to `none` (1).

To create a test suite, use the `generic.GenericObject.configureChildInstanceWithResult` method to create a child of the `sas` object. For a simple example, see `CreateSimpleTestSuite.xml` in Appendix B.

To add tests to a test suite, use `sas.Test.addTest` or `addTests`. To run a test suite, use the `sas.AbstractTest.execute` method. For an example, see `ExecuteTestsFromTestSuite.xml` in Appendix B.

See the Service Test Manager information in the *5620 SAM User Guide* for more information about test suites. See the *5620 SAM-O XML Reference* for more information about the properties and methods of `sas.TestSuite`.

11.5 Generated tests

The 5620 SAM can be used to generate tests using test definitions, policies, and suites. A test definition (subclass of `sas.TestDefinition`), describes the parameters of a test without actually attaching the test to a testing entity. A test policy defines a group of test definitions that apply to a specific type of tested entity; for example, LSPs. By adding a test policy and a list of tested entities to a test suite, tests can be generated for each entity, based on the definitions in the test policy.

For example, by creating a test suite with a group of LSPs and a test policy with an LSP ping test definition, a ping test can be generated for each of the LSPs in the suite.

11.6 Workflow to generate tests

The following workflow lists the high-level steps required to generate tests. See the Service Test Manager information in the *5620 SAM User Guide* for more information about test suites, test policies, and test generation.

- 1 Create a test policy with test definitions. The test policy needs to specify the type of target object to be tested, and the test definitions need to be valid for the object. See the `CreateLSPTestPolicy.xml` file in the code samples. See Appendix B for more information.
- 2 Create a test suite, specifying the test policy. See the `CreateSimpleTestSuite.xml` file in the code samples. Alternatively, you can add your test policy to a test suite. See Appendix B for more information.
- 3 Add target objects to the test suite. See the `AddTestedEntityToTestSuite.xml` file in the code samples. See Appendix B for more information.
- 4 As required, combine steps 2 and 3 to create a test suite and add tested entities to the suite in a single xml request. See the `CreateTestSuite.xml` file in the code samples. See Appendix B for more information.
- 5 Generate the tests:
 - a Use `sas.TestSuite.generateTests`. See the `GenerateTestsFromTestSuite.xml` file in the code samples. See Appendix B for more information.
 - b Use the `generateTests` flag in `sas.TestSuiteAddTestedEntities`. See the *5620 SAM-O XML Reference* for more information and for other methods on `sas.TestSuite`.

11.7 OAM Test Results file retrieval

OSSI clients can use the `<registerSasLogToFile>` method to receive OAM logs that are based on specific class types. Clients must register to receive the logs, and if no client registers, no logs are created.

OAM data is exported to the file after the data is read from the NE. The file is saved to a specified directory on the 5620 SAM server.



Note — The 5620 SAM server verifies the connectivity of the 5620 SAM-O client that is specified in the registerSasLogToFile method using the jmsClientId input parameter. The server deregisters the LogToFile request and the 5620 SAM-O client if the JMS session is disconnected.

Alcatel-Lucent recommends that you run the registerSasLogToFile method every time the 5620 SAM-O client establishes a JMS session.

Table 11-1 describes the input parameters of the <registerSasLogToFile> method.

Table 11-1 <registerSasLogToFile> method input parameters

Input parameter	Description	Values
fullClassName	Package qualified class name in dot separated format.	A comma-separated list of OAM results
compress	Specifies if export files should be compressed when created. A value of true saves files with gzip compression and a filename extension of gz. The default is false. This parameter is optional.	—
continueOnFailure	Specifies whether or not to continue processing requests in this stream if an exception occurs. A value of true continues processing the requests. The default is false. This parameter is optional.	OSSI requests can be sent with this option set to true.
dirName	The relative path to the subdirectory where the log files are to be saved. The path is relative to the OSS XML export directory is /opt/5620sam/server/xml_output	—
jmsClientId	The JMS client ID that is notified when a new export file is created for the specified parameters. A notification message is sent to each client that registers for OAM logs every time an export file is created that is related to that client.	See chapter 4 for more information about JMS client ID formats.
resultFilter	(Optional) Specifies the attributes that are to be included in the exported OAM log. You can use the resultFilter parameter to reduce the size of the export file.	<attribute>attribute_name</attribute>

The OAM log files are created on the 5620 SAM main server. Export file names are unique and are created under the directory that you specify in the registration request. For example:

/opt/5620sam/server/xml_output/oam/<dir_name>

OAM log retrieval example using the <registerSasLogToFile> method

Code 11-1 shows a sample request for an OAM log file using the <registerSasLogToFile> method.

Code 11-1: Sample request to export OAM logs to an XML file using a filter

```
<registerSasLogToFile xmlns="xmlapi_1.0">
  <fullClassName>
    sas.TestResult,
    sas.TraceHop
  </fullClassName>
  <dirName>jmsclientdir</dirName>
  <jmsClientId>oam@1001</jmsClientId>
  <resultFilter>
    <attribute>testSuiteId</attribute>
    <attribute>testId</attribute>
    <attribute>fromNodeId</attribute>
    <attribute>neTestRunIndex</attribute>
    <attribute>resultStatus</attribute>
  </resultFilter>
</registerSasLogToFile>
```

12 – Inventory management

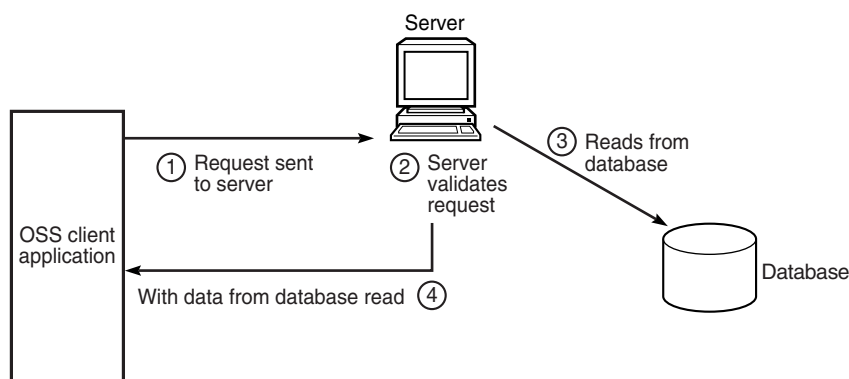
- 12.1 Inventory management overview 12-2**
- 12.2 Inventory retrieval 12-2**
- 12.3 Filtering 12-9**
- 12.4 Network discovery 12-15**
- 12.5 Guidelines for inventory discovery and maintenance 12-16**

12.1 Inventory management overview

Most OSS applications include an inventory management component where information about network elements and their properties are maintained. The 5620 SAM-O interface provides methods that allow OSS applications to send XML API requests to the 5620 SAM database to retrieve the required information about the managed objects and parameters. The inventory can be maintained by keeping it in sync with the 5620 SAM server in real-time, or by periodically sending requests to update the OSS database.

Figure 12-1 shows the message flow of an OSS request for information from the 5620 SAM database.

Figure 12-1 Read from database event flow



17287

12.2 Inventory retrieval

The 5620 SAM-O provides the following methods that can be used to retrieve information from the 5620 SAM network.

- **find**—displays the set of objects of the specified type that match the filter criteria in a streamed XML result
- **findToFile**—displays the set of objects of the specified type that match the filter criteria and places the results in a file on a local or a remote server
- **findFaults**—displays all of the alarms in the network that match the specified filter criteria. See chapter 10 for more information.



Note — Other methods in equipment and network packages that can be used to retrieve specific objects and classes. Alcatel-Lucent does not recommend the use of these methods because they are not optimized.

Table 12-1 describes the XML methods that an OSS client can use to retrieve objects.

Table 12-1 XML methods for object retrieval

Method and parameters	Parameter description
find	
fullClassName	Package qualified class name in dot-separated format
filter (strongly recommended)	Filter on properties of the class corresponding to fullClassName
timeout (optional)	Specifies the time, in ms, after which the operation times out. When this happens, the request is aborted and an exception is returned.
resultFilter (strongly recommended)	Filter for narrowing down the information returned per object
continueOnFailure (optional)	Continues processing requests in the stream if an exception occurs
findToFile	
fullClassName here	Package qualified class name in dot-separated format
fileName	<p>The relative file in which to store the results of this find operation. The format is: [s]ftp://[user:password@]host[:port]/directory/file The following rules must be applied to the format:</p> <ul style="list-style-type: none"> When using a remote client, the directory must be relative to the default FTP directory for the remote client. The characters '/' and ';' are reserved and must be encoded. For example, to specify a file in /tmp directory on the remote client, the path must be encoded as ftp://name@password/%2Ftmp/filename.xml. The path ftp://name@password/tmp/filename.xml specifies a file in a /tmp directory under the default FTP directory of the user. <p>See RFC 1738 Uniform Resource Locators (URL), Section 3.2.2 for more information about the FTP URL path requirements.</p>
timeStamp (optional)	Includes the timestamp in the name of the saved file.
timeout (optional)	Specifies the time, in ms, after which the operation times out.
filter (strongly recommended)	Filter on properties of the class that correspond to fullClassName
resultFilter (strongly recommended)	Filter to refine the displayed information per object
synchronous (optional)	Specifies that the method is run synchronously and that the result is returned only when the operation is complete. The default is true.
continueOnFailure (optional)	Continues processing requests in the stream if an exception occurs
findFaults	
faultFilter	Filter applied to alarms
resultFilter (optional)	Filter to refine the displayed information per object
continueOnFailure (optional)	Continues processing requests in the stream if an exception occurs

Concurrent find or findToFile requests

Up to five <find> or <findToFile> requests from one or more OSSs can be sent. If five concurrent requests are in process, subsequent <find> requests are rejected with an XMLException response, and subsequent <findToFile> requests are rejected with an IMException response.

find method

The find method returns a set of objects of the type specified in the <fullClassName> element that match the filter criteria. The response for the method is in the form of a streamed XML result.

The find method is beneficial for queries of any size because of the minimal system resources required to return query results.



Caution — Network performance can be adversely affected by running numerous queries that gather large amounts of data.

For example, to gather the containment hierarchy of 50 000 services that each have three sites, one access interface per site, and three SDP bindings requires more than 800 Mbytes of space to store the report, which may take many hours to generate.

Code 12-1 shows a sample request to find ACL IP filters.

Code 12-1: Sample request to find ACL IP filters

```
<SOAP:Body>
  <find xmlns="xmlapi_1.0">
    <fullClassName>aclfilter.IpFilter</fullClassName>
    <filter>
      <equal name="objectFullName" value="IP Filter:1000"/>
    </filter>
  </find>
</SOAP:Body>
```

findToFile method

The findToFile method returns a set of objects of the type specified in the <fullClassName> element that match the filter criteria. You can use the following options to save the results:

- record the results in a file on a 5620 SAM server
- transfer the results by FTP to a remote server

Because the results of the request are written to a file, the XML response that is sent to the OSS from the findToFile request is similar to Code 12-2. The Code indicates that the request was successful and that the file is available.

Code 12-2: Sample findToFileResponse

```
<findToFileResponse xmlns="xmlapi_1.0">
```

The OSS applications provide an option to receive a JMS message that will alert when the file is available if the OSS application listens to the JMS message feed on the 5620 SAM main server. The JMS FileAvailableEvent is sent when all of the results are written to a file. The message header identifies the client that made the request, and the message includes the URI of the result and the request ID. If the request is not completed, an exception appears to indicate that a problem has occurred will result. See “FileAvailableEvent” in chapter 4 for more information.

The <findToFile> method is especially useful for large queries because of the minimal system resources are required to generate query results.

Code 12-3 shows a sample request to retrieve historical statistics from an NE.

Code 12-3: Sample request to retrieve historical statistics from an NE

```
<SOAP:Body>
  <findToFile xmlns="xmalapi_1.0">
    <fullClassName>equipment.InterfaceAdditionalStatsLogRecord</full
ClassName>
    <filter>
      <and>
        <equal name="monitoredObjectPointer"
value="network:10.1.202.93:shelf-1:cardSlot-1:card:daughterCardSlot-1:daught
erCard:port-3"/>
        <between name="timeCaptured" first="1127142900000"
second="1127143800000"/>
      </and>
    </filter>
    <fileName>Equipment.InterfaceAdditionalStatsLogRecord.xml</fileN
ame>
  </findToFile>
</SOAP:Body>
```

File format of findToFile results

The file format of a <findToFile> results data file are XML formatted similar to the streaming results that are returned from a regular <find> request. The actual objects returned will also depend on which result filters are specified in the request. See section 12.3 for information about result filters.

Local file storage of query results

The URL used to record the findToFile method responses is a relative path on the 5620 SAM main server. The path root is specified as the XML output directory during 5620 SAM server installation.



Note — You can manage the volume of files that are stored locally by configuring the following attributes in the xmalapi section of the nms/config/nms-server.xml file on the 5620 SAM main server:

- xmlTimeToKeepFile—Specifies how long, in m, to keep each file created by findToFile requests. The range is 5 to 1440. The default is 15.
 - xmlCheckDiskInterval—Specifies how often, in m, to check the file system for file purging. The range is 1 to 60. The default is 5.
- Code 12-4 shows an example storage configuration in nms-server.xml.

Code 12-4: Sample file storage configuration

```
<xmlapi
....
....
xmlTimeToKeepFile="1440"
xmlCheckDiskInterval="60" />
```

Code 12-5 shows an example of how to record the results to a file.

Code 12-5: Sample configuration of local file storage

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">

  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <security>
        <user>user_name</user>
        <password>MD5-hashed_password</password>
      </security>
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>

  <SOAP:Body>
    <findToFile xmlns="xmlapi_1.0">
      <fullClassName>equipment.PhysicalPort</fullClassName>
      <filter>

<and>
      <equal name="siteId" value="10.1.202.93"/>
      <equal name="shelfId" value="1"/>
      <equal name="cardSlotId" value="1"/>
      <equal name="daughterCardSlotId" value="1"/>
      <equal name="mode" value="access"/>
</and>

      </filter>
      <fileName>equipmentPhysicalPort.xml</fileName>
    </findToFile>
  </SOAP:Body>
</SOAP:Envelope>
```

Remote file storage of query results

You can configure the findToFile method to record the query results to a file on a remote workstation using FTP. The fileName parameter of the findToFile method allows you to configure the optional FTP credentials, host identifier, and file path. If the FTP credentials are not specified, the 5620 SAM-O uses:

- anonymous, as the FTP username
- an e-mail address, as the FTP password

Code 12-6 is an example remote storage configuration for the findToFile method.

Code 12-6: Sample configuration of remote file storage

```
<findToFile xmlns="xmlapi_1.0">
  <synchronous>false</synchronous>
  <fullClassName>class_name</fullClassName>
  <fileName>
[s]ftp://[user[:password]@]host[:port]/directory/file
  </fileName>
  <resultFilter>
```

```
.  
.   
.   
    </resultFilter>  
</findToFile>
```

FTP file permission configuration

Files created using the findToFile method have the file permissions of the 5620 SAM-O process. The OSS clients that FTP to the 5620 SAM server to obtain the files need read and write permissions for the directory of the files.



Note — The findToFile method returns the relative path for the request. You must make sure that the root directory of your FTP application aligns with the root directory used by the 5620 SAM server to store findToFile results, which is *path/xml_output*, where *path* is the 5620 SAM server installation location, typically */opt/5620sam/server*.

Procedure 12-1 describes how to configure the permissions for the findToFile FTP directory and FTP account.

Procedure 12-1 To configure remote findToFile result storage

Perform this procedure to configure FTP file storage for findToFile results.

- 1 For each 5620 SAM-O client application, create a directory on the remote station below the directory associated with the findToFile method. In the following example, the findToFile method uses a directory called 5620SAM-O, and there are three client applications:

```
.../5620SAM-O/clientapp1  
.../5620SAM-O/clientapp2  
.../5620SAM-O/clientapp3
```

- 2 Enable read and write permissions for each directory created in step 1.
- 3 Create an FTP account for use by the 5620 SAM-O clients. The FTP account home directory must be the directory associated with the findToFile method, for example, 5620SAM-O.
- 4 Specify the appropriate client directory in the <fileName> section of the findToFile request for each client, as shown in the following example, which uses a directory named in step 1:

```
<fileName>  
  
[s]ftp://[user[:password]@]host[:port]/clientapp1/output_file  
  
</fileName>
```

Synchronous and asynchronous requests

By default, the findToFile method runs synchronously from the invocation of the request. Any errors during the query or file creation results in a JMS notification. The 5620 SAM-O returns the following notifications for findToFile problems:

- a JMS notification for an error that occurs during the query or during file creation
- an error message, if the number of findToFile requests exceeds the limit configured in the nms-server.xml file for the pool or queue value

The <synchronous> element in a findToFile request specifies the execution type for the method. When synchronous requests are used, the request blocks further requests until the current findToFile request is complete. You can set the <synchronous> element to false to run requests asynchronously.

Scheduled <findToFile>

The <findToFile> method supports the scheduled export of data using the 5620 SAM scheduler framework. The options that are supported by the scheduler framework are available for scheduling a <findToFile> task. Table 12-2 describes how to schedule a data export.

Table 12-2 Creating a scheduled data export

Task	Example
1. Create a schedule or choose an existing schedule.	<pre> <generic.GenericObject.configureChildInstance xmlns="xmlapi_1.0"> <deployer>immediate</deployer> <distinguishedName>scheduleManager</distinguishedName> <childConfigInfo> <schedule.SamSchedule> <actionMask> <bit>create</bit> </actionMask> <id>10</id> </schedule.SamSchedule> <displayName>samSchedule-10</displayName> <description>samSchedule-10</description> <frequency>minute</frequency> <onGoing>true</onGoing> <runEveryMinute>5</runEveryMinute> </childConfigInfo> </generic.GenericObject.configureChildInstance> </pre>

(1 of 2)

Task	Example
2. Create a findToFileTask or reference an existing task.	<pre> <generic.GenericObject.configureChildInstance xmlns="xmlapi_1.0"> <deployer>immediate</deployer> <distinguishedName>scheduleManager</distinguishedName> <childConfigInfo> <schedule.OssTask> <actionMask> <bit>create</bit> </actionMask> <id>10</id> </schedule.OssTask> </childConfigInfo> <displayedName>FindToFile</displayedName> <description>FindToFile</description> <commandName>FindToFile</commandName> <command> <findToFile xmlns="xmlapi_1.0"> <fullClassName>netw.NetworkElement</fullClassName> <fileName>netw.xml</fileName> <resultFilter> <children> <resultFilter class="equipment.Equipment"> </resultFilter> </children> </resultFilter> </findToFile> </command> </command> </generic.GenericObject.configureChildInstance> </pre>
3. Create a samScheduledTask, which points to a schedule, and a findToFile task.	<pre> <generic.GenericObject.configureChildInstance xmlns="xmlapi_1.0"> <deployer>immediate</deployer> <distinguishedName>scheduleManager</distinguishedName> <childConfigInfo> <schedule.SamScheduledTask> <actionMask> <bit>create</bit> </actionMask> <id>15</id> <administrativeState>2</administrativeState> <displayedName>FindToFileTask-15</displayedName> <description>FindToFileTask15</description> <scheduledObjectPointer>scheduleManager:OssTask-111</scheduledObjectPointer> <schedulePointer>scheduleManager:samSchedule-10</schedulePointer> </schedule.SamScheduledTask> </childConfigInfo> </generic.GenericObject.configureChildInstance> </pre>
4. Turn up, shut down, start or stop the samScheduledTask.	—

(2 of 2)

12.3 Filtering

You can define a filter for any object property to limit or refine the following:

- information returned to the OSS application after a request is made to the server
- configuration requests sent to the server

The filter syntax is part of an XML OSS request. In code 12-7, the filter specifies that the server executes the method when the `monitoredObjectSiteId` element has an IP address of 1.11.111.111. The server executes the request and returns information from the NE at IP address of 1.11.111.111.

Code 12-7: Sample filter syntax

```
<filter>
  <equal name="monitoredObjectSiteId" value="1.11.111.111" />
</filter>
```

Filter construction requires that you know the class to which you want to apply the filter. The class is defined by the class attribute in the top-level `<filter>` element. Code 12-8 shows an example of using the class attribute to define the filter class.



Note — The use of filters when retrieving deployer information is handled slightly differently from using the `find` or `findToFile` methods. See section 14.3 for more information about using deployer methods.

Code 12-8: Sample filter using class attribute

```
<filter>
  <equal class="netw.NetworkElement".../>
</filter>
```

Children filters

The `find` and `findToFile` methods support the use of nested children filters for an object. Code 12-9 shows the use of a filter that returns VPLS sites with the following criteria:

- displayed name starts with *MySites*
- contains children whose displayed name starts with *MyInterfaces*
- contains *ressubscr.SapSubMgmtCfg* with disabled administrative status

The attribute *childClass* is required if the attributes in the children filter are specific to a child class. You can set the *withChildrenOnly* attribute to true to specify that an object is to be returned only if it has children that match the child filter. The *withChildrenOnly* attribute is optional and is set to false by default.

Code 12-9: Children filters

```
<filter class="vpls.Site">
  <wildcard name="displayName"
    value="MySites%"/>
  <children>
    <filter class="vpls.Site" withChildrenOnly="true">
      <wildcard name="displayName"
        value="MyInterfaces%"/>
      <children>
        <filter class="vpls.L2AccessInterface"
          childClass="ressubscr.SapSubMgmtCfg" withChildrenOnly="true">
          <equal name="adminStatus" value="disabled"/>
        </filter>
      </children>
    </filter>
  </children>
</filter>
```


Filter element types

The 5620 SAM-O supports the following filter element types:

- composite—a Boolean filter that contains leaf elements, other composite elements, or both
- leaf—a filter based on an arithmetic or string function
- result—applies only to returned objects

Composite filter

Composite filter elements have associated child elements. Table 12-3 describes the composite filter element types.

Table 12-3 Composite filter element types

Filter element	Description	Example
and	AND the results of two containing filter elements	X and Y
or	OR the results of two containing filter elements	X or Y
not	NOT the results of one containing filter elements	NOT X

Code 12-10 shows a sample filter to yield results that are the same as the given value. You must specify the bit mask value using a decimal integer format.

Code 12-10: Sample filter using equal and bitmask

```
<filter>
  <and>
    <equal name="siteId" value="10.1.1.52"/>
    <and>
      <anyBit name="family" value="64"/>
      <not>
        <equal name="localAS" value="20"/>
      </not>
    </and>
  </and>
</filter>
```

Leaf filter

Leaf filter elements do not have associated child elements. Table 12-4 describes the leaf filter element types.

Table 12-4 Leaf filter element types

Filter element	Description	Example
equal	Return true if the value is the same as the given value.	<equal name="localAS" value="20"/>
notEqual	Return true if the value is not the same as the given value.	<notEqual name="localAS" value="20"/>

(1 of 2)

Filter element	Description	Example
greater	Return true if the numeric value is greater than the given value.	<greater name="localAS" value="20" />
greaterOrEqual	Return true if the numeric value is greater than or equal to the given value.	<greaterOrEqual name="localAS" value="20" />
less	Return true if the numeric value is less than the given value.	<less name="localAS" value="20" />
lessOrEqual	Return true if the numeric value is less than or equal to the given value.	<lessOrEqual name="localAS" value="20" />
anyBit	Return true if the value bit ANDed with the given value does not return 0. The filter is true if any of the "1" bits in the specified value are enabled in the property value. This only applies to non-negative numeric types and bitmask types.	<anyBit name="parameter" value="3" />
allBits	Return true if the value bit ANDed with the given value returns the given value. The filter is true if all of the "1" bits in the specified value are enabled in the property value. This only applies to non-negative numeric types and bitmask types.	<allBits name="parameter" value="5" />
wildcard	Perform equality using wildcards. The character % signifies 0 or more characters of any type, and the character _ signifies one character of any type.	<wildcard name="SiteId" value="1.%.%.%" />
in	Return true if a numeric or string value equals one of the values in the given comma-separated list.	<in name="parameter" value="[Edge_14, Aggregation_14, Core_14]" />
subset	Return true if a string value is a subset of the given string.	<subset class="service.Site" name="tier" value="123" />
superset	Return true if a string value contains the given string.	<superset class="service.Site" name="siteName" value="Core_" />
notSuperset	Return true if a string value does not contain the given string.	<notSuperset class="service.Site" name="siteName" value="Edge_" />
between	Return true if the numeric value is between the first and second given values	<between name="preference" first="150" second="200" />
past	Return true if a time value is greater than the given time.	<past time="1170864259082" />
empty	Match everything; this is the same as specifying nothing between the <filter> and </filter> tags.	<empty />

(2 of 2)

Code 12-11 shows an example of the numeric value that is greater than or equal to the given value. You can specify the value of an enumeration using a numeric or string format.

Code 12-11: Sample filter example - usage of greater, between, and equal

```
<filter>
  <and>
    <greater name="localAS" value="0" />
```

```

        <and>
          <equal name="domain" value="network"/>
          <between name="preference" first="150" second="200" />
        </and>
      </and>
    </filter>

```

Composite and leaf filters can be used together for not only the top level object, but also for children. For children filters, you must specify the class of the child object being filtered. For example, `<filter class="service.Site">`.

Code 12-12 shows an example of composite and leaf filters used together for a top level `epipe.Site` object and children.

Code 12-12: Sample filter example - composite and leaf filters used together for a top level object and children

```

<find xmlns="xmlapi_1.0">
  <fullClassName>service.Service</fullClassName>
  <filter>
    <wildcard name="displayName"
              value="%Epipe%"/>
    <children>
      <filter class="service.Service">
        <wildcard name="displayName"
                  value="%epipe%"/>
        <children>
          <filter class="service.Site">
            <wildcard name="description"
                      value="%Epipe%"/>
          </filter>
        </children>
      </filter>
    </children>
  </filter>
  <resultFilter>
    ...
  </resultFilter>
</find>

```

Result filter

A result filter applies only to the returned objects. Table 12-5 describes the result filter element types.

Table 12-5 Result filter element types

Children values	Description	Example
<resultFilter>	<p>You can use the <resultFilter> element to define the attributes and children returned for 5620 SAM-O queries. The 5620 SAM-O returns all attributes and children if you do not specify the types in the <resultFilter> element.</p> <p>A nested <resultFilter> element can contain the class attribute that specifies the following filtering types:</p> <ul style="list-style-type: none"> Package-qualified class name of the returned child; for example, <resultFilter class="equipment.Port"> Generic base class name for all managed objects; for example, <resultFilter [class="{child class name ManagedObject}"]> <p>Some objects have their children suppressed by default; for example, <childrenSummaryEnabled>.</p>	—
<attribute>	Attributes on the managed objects for which you are searching; for example, <attribute>objectFullName</attribute>	—
<children>	<p>Signifies a list of children and their associated attributes defined by the sub-result filters. An empty tag, <children/>, signifies the return of no children past this level. No <children> tag returns all children.</p> <p>The recursive attribute allows you to filter the children using the same rules as the parent; for example, <children recursive="yes"/>. You do not need to use nested <resultFilter> elements if you enable the recursive attribute.</p>	—

Code 12-13 shows an example of a result filter with contained children. There is no limit to the depth of the contained result filters and children tags.

Code 12-13: Result filter with contained children

```

<resultFilter>
  <attribute>attribute</attribute>
  <attribute>status</attribute>
  .
  .
  .
  <children [recursive="{yes|no}"]>
    <resultFilter [class="{child class name|ManagedObject}"]>
      <attribute>attribute</attribute>
    </resultFilter>
  .
  .
  .
</children>
</resultFilter>

```

Code 12-14 shows an example configuration of a result filter that returns the administrative status and the port speed for the network equipment.

Code 12-14: Result filter for equipment

```

<SOAP:Body>
  <find xmlns="xmlapi_1.0">
    <fullClassName>equipment.PhysicalPort</fullClassName>
    <filter>
      <equal name="siteName" value="sim202_93"/>
    </filter>
    <resultFilter>
      <attribute>administrativeState</attribute>

```

```

        <attribute>speed</attribute>
      </children>
    </resultFilter>
  </find>
</SOAP:Body>

```

Code 12-15 shows an example configuration of a result filter in a request that uses the `generic.GenericObject.configureInstanceWithResult` method. See section 14.2 for more information about the generic methods that are applicable to all objects.

Code 12-15: Result filter in a request that uses generic methods

```

<SOAP:Body>
  <generic.GenericObject.configureInstanceWithResult
xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <!-- Network Interface -->
    <distinguishedName>network:10.1.1.223:router-1:ip-interface-5</d
istinguishedName>
    <includeChildren>false</includeChildren>
    <configInfo>
      <rtr.NetworkInterface>
        <actionMask><bit>modify</bit></actionMask>
        <!-- vRtrIfUp/vRtrIfDown -->
        <administrativeState>vRtrIfUp</administrativeState>
      </rtr.NetworkInterface>
    </configInfo>
    <resultFilter>
      <attribute>objectFullName</attribute>
    </resultFilter>
    <attribute>displayedName</attribute>
    <children recursive="yes" />
  </generic.GenericObject.configureInstanceWithResult>
</SOAP:Body>

```

12.4 Network discovery

5620 SAM object hierarchy

For OSS inventory management applications, OSS developers need to identify the network elements and attributes that are required, the structure of the inventory model, and the relationships amongst the objects. For example, the database of a typical inventory management system includes the physical elements of the routers such as cards, ports and links, and the association with the services and service entities.

Based on the OSS requirements, a OSS inventory application can extract specific classes and parameters that are defined in the 5620 SAM-O model from the 5620 SAM-managed network. The 5620 SAM models the following object types in the network:

- physical elements, such as routers (sites), shelves, cards, ports, and physical links
- logical elements, such as protocols, policies, services and service entities, and OAM

The `<objectFullName>` tag displays the hierarchy of how the objects are organized and used when XML requests are built. The following is an example for a port:

```

network:<siteID>:shelf-<ID>:cardSlot-<ID>:card:daughterCardSlot-<ID>
:daughterCard:port-<ID>

```

From the above tag, you can see the containment as follows:

```
network <siteId>
  |----> shelf <ID>
    |----> cardSlot <ID>
      |-----> card:daughterCardSlot <ID>
        |-----> daughterCard:port <ID>
```

The 5620 SAM-O object model is organized in a hierarchy and has parent-child relationships that are described for all of the management objects in both the *5620 SAM-O XML Reference* (see chapter 7) and schema files (see chapter 8).

12.5 Guidelines for inventory discovery and maintenance

Most OSS applications perform network inventory management. Alcatel-Lucent recommends the following inventory management guidelines for an OSS:

- During startup, perform an initial discovery of the network and maintain a database that includes all of the required data about network object.
- For the find and findToFile methods, use <filter> and <resultFilter> to reduce the number of objects that are returned and improve the retrieval rates.
- Include related find requests in one HTTP connection for each find request.
- Minimize the number of find requests that are sent at the same time.
- Use the event-driven nature of the JMS interface to listen for notifications to synchronize the database when changes occur in the network (instead of a series of scheduled find or findToFile requests).
- When you use the findToFile method, configure the file management attributes, if required. See “[Local file storage of query results](#)” in section 12.2 for more information about managing the volume of files that are stored locally on the 5620 SAM server.
- When an OSS inventory database is updated with JMS messages, verify at the system initial discovery or as part of recovering from loss connectivity that all of the incoming JMS messages are buffered during a full network discovery. See the Procedure 4-1 for more information about resynchronizing an OSS database with the 5620 SAM.

13 – Accounting and performance monitoring

- 13.1 Accounting and performance monitoring overview 13-2
- 13.2 References 13-2
- 13.3 Key packages and classes 13-2
- 13.4 Statistic objects 13-3
- 13.5 Workflow to retrieve statistics data 13-4
- 13.6 Scheduled and on-demand statistics 13-4
- 13.7 Statistics retrieval methods 13-6
- 13.8 Statistics retrieval using registerLogToFile 13-6
- 13.9 Statistics retrieval using findToFile 13-10
- 13.10 Statistics monitoring using JMS 13-14
- 13.11 Collecting JMS performance statistics 13-14
- 13.12 Sample request to retrieve historical statistics from an NE 13-15
- 13.13 Sample request to collect interface statistics 13-15

13.1 Accounting and performance monitoring overview

The 5620 SAM provides a scalable platform for reliably collecting statistics from the managed NEs and the 5620 SAM system. The statistics are typically used for monitoring and troubleshooting a 5620 SAM network, and for SLA and billing functions.

The 5620 SAM can collect the following statistics:

- performance statistics—collected by polling NE MIBs, transferred to the 5620 SAM using SNMP and stored in the 5620 SAM database
- accounting statistics—collected in files on NEs, transferred to the 5620 SAM using FTP or SCP and stored in the 5620 SAM database
- server performance statistics—collected from the 5620 SAM system functions and processes, and stored in the 5620 SAM database

To collect statistics, the 5620 SAM requires policies that specify the following:

- the network or service objects from which to collect statistics
- the statistics counters to collect
- the collection rate
- how long the 5620 SAM is to retain the collected statistics data

13.2 References

The following references are of interest to an OSS developer working in the accounting and performance monitoring domain:

- *5620 SAM Statistics Management Guide*— useful to an OSS developer during both the design phase (to establish an understanding of 5620 SAM statistics) and implementation and testing (to configure the 5620 SAM for statistics collection). This guide also includes a comprehensive list of statistics counters on all supported Alcatel-Lucent nodes.
- *5620 SAM User Guide*—application assurance accounting statistics are documented in this guide.
- *5620 SAM-O XML Reference*—documents specific classes, attributes, and methods used to configure and collect statistics using the 5620 SAM-O interface. See section 13.3 for the packages and classes that are of interest to an accounting and performance OSS developer.

13.3 Key packages and classes

log package

The log package contains the `CurrentData` and `LogRecord` abstract classes, which encapsulate collected statistics. The concrete classes that inherit from these are in the same packages as the target objects that statistics are being collected against. For example, access egress octets collected on a service will be in a `service.AccessEgressOctetsLogRecord`, which inherits from `log.LogRecord`.

accounting package

The accounting package contains methods that you can use to define accounting policies for accounting statistics collection.

file package

The file package contains methods that you can use to define file policies. File policies define the storage location, retention period, and rollover period for accounting statistics files on managed devices.

root package

The root package contains the `findToFile` and `registerLogToFile/deregisterLogToFile` methods used to retrieve statistics data. See sections [“Statistics retrieval using registerLogToFile”](#) and [“Statistics retrieval using findToFile”](#) for more information. See general methods and types information in the *5620 SAM-O XML Reference* for information about these methods.

13.4 Statistic objects

The 5620 SAM-O uses the following objects to track statistics:

- current data record, for the most recent performance statistics
- log records, for historical accounting and performance statistics

Log records are maintained for scheduled and on-demand statistics. The 5620 SAM-O organizes statistics objects by package. For example, the `bgp` package defines the BGP statistics. You must use the corresponding log record object to retrieve statistics for an object. For example, the BGP peer statistics are tracked by the `PeerStatsLogRecord` class in the `bgp` package. The log records contain the following information:

- time captured
- monitored object identifier
- relevant statistics

A policy is associated with each type of log record. You can use the log policy to modify the thresholds and maximum log size that are associated with a log record. For example, you can modify the `PeerStatsLogPolicy` class to customize the log records collected for the BGP Peer Stats. The default settings generally provide appropriate values for the thresholds and log size.



Note — 5620 SAM system performance can be adversely affected by increasing the number of historical statistics entries recorded by the 5620 SAM. Network performance degradations include increased time to list log records from the GUI and OSS clients, increased size of Oracle tablespaces, and increased time to backup the database.

Alcatel-Lucent recommends the efficient management of the number of historical statistics recorded by the 5620 SAM to optimize network performance.

13.5 Workflow to retrieve statistics data

Use the 5620 SAM-O, and the registerLogToFile method or the findToFile method, to transfer statistics log records from the 5620 SAM database to an OSS client application.

- 1 Construct a SOAP request to perform one of the following:
 - a Use the findToFile method to retrieve specific accounting or performance statistics.
 - i Specify the log record name.
 - ii Specify an XML export format.
 - iii Specify the name of the exported data file.
 - b Use the registerLogToFile method to create accounting or performance statistics export files.
 - i Specify the statistics class or classes.
 - ii Specify a location for the exported data files.
 - iii Specify whether to compress the files.
 - iv Specify the attributes to include in the exported data records.
- 2 Specify a filter to limit the data stored to the file.
- 3 Send the request.
- 4 Receive a response or exception to the request.

See the following for more information about configuring statistics collection:

- [“Sample request to retrieve historical statistics from an NE”](#)
- [“Sample request to collect interface statistics”](#)
- [“OAM log retrieval example using the <registerSasLogToFile> method”](#)

13.6 Scheduled and on-demand statistics

You can use the 5620 SAM-O to collect statistics data using the following methods:

- scheduled polls
- on-demand collection

Scheduled and on-demand performance statistics use current data records. History is maintained for the scheduled polls using the LogRecord type. No history is maintained for on-demand statistics. The data from the last scheduled poll is also stored using the LogRecord type. For example, equipment.InterfaceAdditionalStats are collected in a 5-minute polling interval. The scheduled polling creates a current data record and a log record.



Note — The current data records do not apply to accounting statistics.

The <objectFullName> of the scheduled current data starts with: logger:scheduled. The on-demand current data <objectFullName> starts with: logger:real-time. A historical log record is also created for the scheduled current data. See “[Object full name](#)” in section 8.1 for more information about the unique identifier of the instance of the object.

If you initiate a non-scheduled collection using the generic.GenericObject.triggerCollect method, a new on-demand current data object is brought into the system. The 5620 SAM-O does not modify the scheduled current data object or create a new historical log record. On-demand statistics are propagated to the history as non-scheduled statistics and do not affect any scheduled statistic retrievals.

You can list several instanceNames when you use the generic.GenericObject.triggerCollect method for a non-scheduled collection, but you must specify the current data class for each of the instances that contain the statistics that you want to collect. The current data classes must be listed in currentDataClasses. Code 13-1 shows a sample non-scheduled collection.

Code 13-1: Sample non-scheduled statistics collection

```
<generic.GenericObject.triggerCollect xmlns="xmlapi_1.0">
<instanceNames><string>network:IP_address:shelf-1:cardSlot-1:card:daughterCardSlot-1:daughterCard:port-4</string></instanceNames>
<currentDataClasses><string>equipment.InterfaceAdditionalStats</string></currentDataClasses>
</generic.GenericObject.triggerCollect>
```

For the equivalent behavior of the Collect button on the 5620 SAM GUI, you can specify one instance in the list with the corresponding data class in the second list. For the equivalent behavior of the Collect All button on the 5620 SAM GUI, you can specify all relevant instances in the list with all corresponding data classes in the second list. See the *5620 SAM User Guide* for more information about how to use the Collect and Collect All buttons.

13.7 Statistics retrieval methods

You can use the 5620 SAM-O to collect statistical data using the following methods:

- registerLogToFile to continually collect accounting and performance statistics
- findToFile to occasionally collect specific accounting or performance statistics
- monitor statistics events using JMS (deprecated)



Note — The 5620 SAM allows you to configure queue filters for accounting statistics. See the *5620 SAM Statistics Management Guide* for more information.

Procedure 13-1 To configure the registerLogToFile storage requirements

Perform this procedure to specify the storage parameters for exported statistics files on a 5620 SAM server.

- 1 Choose Administration→System Preferences from the 5620 SAM main menu. The System Preferences form opens.
 - 2 Click on the Statistics tab button.
 - 3 Configure the parameters:
 - Log Retention Time (minutes)—specifies how long, in m, a 5620 SAM server retains each exported file; applies to accounting and performance statistics
The parameter range is 5 to 600; the default is 15.
 - Log Rollover Time (minutes)—specifies how long, in m, the 5620 SAM keeps a file open to accumulate statistics data; applies to performance statistics only
The parameter range is 5 to 600; the default is 15.
 - 4 Click on the OK button. A dialog box appears.
 - 5 Click on the Yes button. The System Preferences form closes.
-

13.8 Statistics retrieval using registerLogToFile

You can use the registerLogToFile method to create accounting or performance statistics data files for specific classes. When the registerLogToFile method is used, the 5620 SAM stores the specified statistics data in the database. It also generates files on the 5620 SAM server with the requested data for each registered 5620 SAM-O client. These two activities are done in parallel, so data is available in files before it is available for export from the database.

A LogFileAvailableEvent is generated each time a file is ready for retrieval by an OSS client. See chapter 4 for information about JMS events.

You can specify multiple accounting or performance statistics classes in a registration request. Before you can use the registerLogToFile method to export performance statistics, you must create a MIB policy to enable collection of the required statistics.



Note 1 – Alcatel-Lucent recommends using the registerLogToFile method for ongoing statistics retrieval, and the findToFile method only for occasional accounting or performance statistics queries. Using the findToFile method is not recommended in large networks with high statistics collection rates. See [“Statistics retrieval using findToFile”](#) in this chapter for information about using the findToFile method.

Note 2 – The use of the registerLogToFile file compression option requires additional 5620 SAM server resources. The additional load may impair the 5620 SAM ability to collect the maximum number of statistics records documented in the *5620 SAM Planning Guide*.

Note 3 – The 5620 SAM verifies 5620 SAM-O client connectivity; it deregisters a registerLogToFile request and the associated 5620 SAM-O client if the JMS session is disconnected.

Alcatel-Lucent recommends that you run the registerLogToFile method every time a 5620 SAM-O client establishes a JMS session.

The span of control associated with the JMS client is applied to each registerLogToFile request only at the NE level. If an NE is not within the client span of control, the 5620 SAM does not include the requested statistics for the NE in the exported files.

Table 13-1 describes the input parameters of the registerLogToFile method.

Table 13-1 registerLogToFile method input parameters

Input parameter	Description	Values
fullClassName	Package qualified class name in dot-separated format	A comma-separated list of accounting statistics classes, performance statistics classes, or both
dirName	The relative path of the directory that is to contain the exported files. The path is relative to the accountingStats or performanceStats directory below the XML output directory specified during 5620 SAM server installation or upgrade. Alcatel-Lucent recommends using a separate directory for each application that requires exported statistics.	<i>string</i>
compress	Optional parameter that specifies whether data files are compressed as they are exported. Setting this parameter to true may affect 5620 SAM statistics collection performance.	The options are: <ul style="list-style-type: none"> true—compresses data files using gzip; each exported file has a .gz name extension false (default)—does not compress exported files

(1 of 2)

Input parameter	Description	Values
jmsClientId	The JMS client ID that is to be notified when a new export file is ready for retrieval	<i>string</i> See chapter 4 for information about JMS client ID formats.
resultFilter	Optional parameter that specifies the attributes to include in the exported data records; you can use the parameter to reduce the size of the exported files	<attribute>attribute_name</attribute>
continuoOnFailure	Optional parameter that specifies whether to continue processing requests in the stream after an exception occurs	The options are: <ul style="list-style-type: none"> • true—processing continues after an exception • false (default)—processing halts after an exception

(2 of 2)

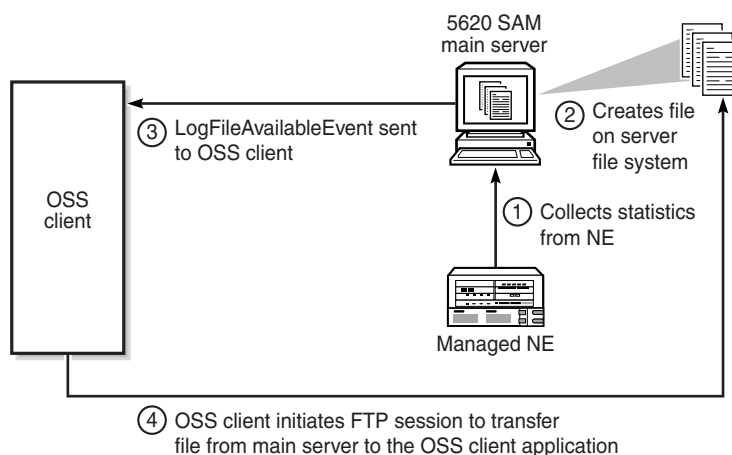
See registerLogToFile.xml in Appendix B for a working example.

The statistics export files are created on the 5620 SAM main or auxiliary server that performs the statistics collection. The name of each exported file is created using the requesting client ID and the current timestamp, so is unique. A 5620 SAM-O client can retrieve the exported files using FTP. Alcatel-Lucent recommends that you configure the OSS client FTP account on each main and auxiliary server to use the same credentials. This ensures that file retrieval is not significantly interrupted in the event of a disruption such as a server activity switch.



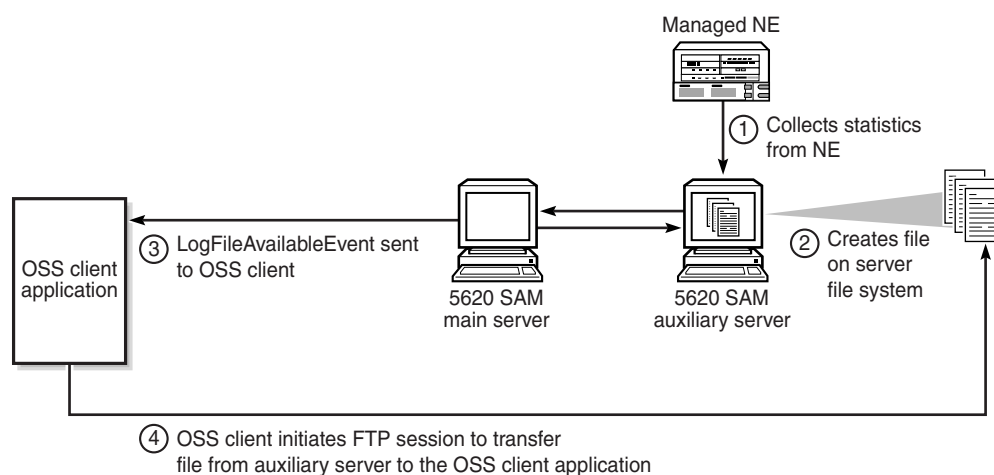
Note — A statistics file export can generate a large number of files and consume considerable disk space. By default, the exported files are deleted every 15 minutes. You can manage the disk space that the files consume using the 5620 SAM GUI. See Procedure 13-1 for more information.

Figure 13-1 shows the process associated with the event-driven statistics export from a 5620 SAM main server after a 5620 SAM-O client registers for notifications using the registerLogToFile method.

Figure 13-1 Event-driven registerLogToFile process using 5620 SAM main server

19436

Figure 13-2 shows the process associated with the event-driven statistics export from a 5620 SAM auxiliary server after a 5620 SAM-O client registers for notifications using the registerLogToFile method.

Figure 13-2 Event-driven registerLogToFile process using 5620 SAM auxiliary server

19437

A 5620 SAM-O client can deregister from receiving notifications for accounting statistics and stop the export file creation by sending a deregisterLogToFile request. A deregisterLogToFile request requires only the JMS client ID as a parameter. See registerLogToFile.xml in Appendix B for a working example.

A 5620 SAM-O client is also deregistered when the client JMS session is closed. The client must register again to receive notifications.

Statistics recovery after OSS connection loss

If an OSS client loses connectivity to the 5620 SAM main or auxiliary server that is collecting statistics, the OSS needs to retrieve the registerLogToFile export files when it reconnects. The files are still available for retrieval if the log retention time does not elapse before the client reconnects. See Procedure 13-1 for information about configuring the log retention time.

Statistics recovery after JMS client unsubscribed

If an OSS client JMS subscription becomes unsubscribed, the 5620 SAM server deregisters that registerLogToFile session and stops creating export files after a configured time interval, which is 20m by default. Contact an Alcatel-Lucent technical-support representative for information about how to configure this time interval.

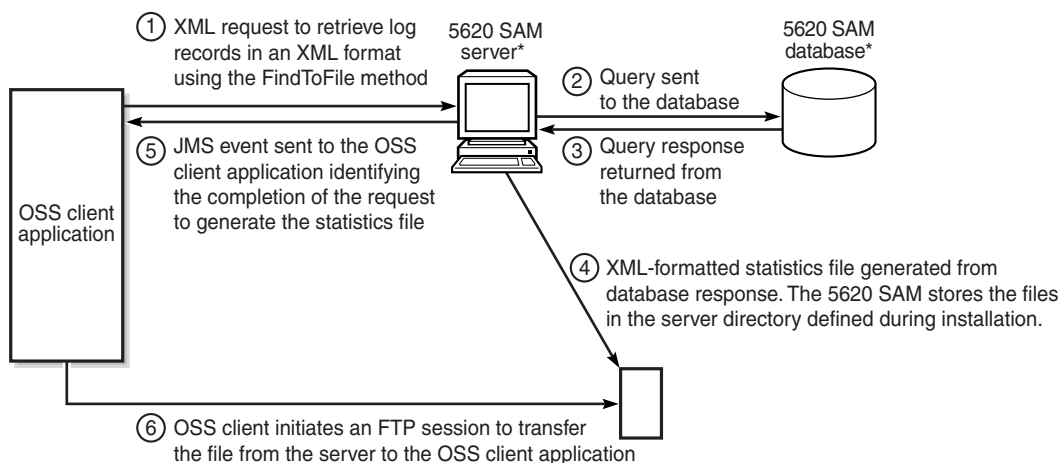


Note — Alcatel-Lucent does not recommend the use of findToFile to retrieve missed registerLogToFile accounting statistics log records. Multiple findToFile requests may create an excessive server load and degrade 5620 SAM performance.

13.9 Statistics retrieval using findToFile

You can use the findToFile method to transfer accounting and performance statistics log records from the 5620 SAM database to your OSS client application. The findToFile method can save statistics from multiple NEs to one file. The 5620 SAM sends a FileAvailableEvent message each time a statistics file becomes available. See chapter 4 for information about JMS events. Figure 13-3 shows the process associated with the statistics retrieval using the findToFile method.

Figure 13-3 findToFile statistics retrieval process



* The 5620 SAM server and database can be installed on the same or separate stations

18007

See chapter 12 for more information about the findToFile method.



Note 1 — In addition to the illustrated scenario, step 4 of figure 13-3 can result in the following:

- if a file name is specified, a write file is created
- if an FTP address is specified, the 5620 SAM server sends the file to a URL without creating an intermediate file
- if the file location is on a remote disk that is read directly by the client application, the application may read or copy the file.

The OSS application is responsible for managing files and related disk space associated with these requests.

Note 2 — The 5620 SAM maps the object FDNs to a corresponding database index. Alcatel-Lucent recommends using FDN properties in queries to maximize the processing speed of the query. The 5620 SAM-O XML API queries should also use concrete classes rather than an abstract class to optimize performance.

Note 3 — There are two methods to export statistics from the 5620 SAM— registerLogToFile and findToFile. The registerLogToFile method is recommended to minimize collection latency and to reduce system load. For situations where fewer than 400 000 statistics records are retrieved in 15 minutes and where longer collection latency is tolerable, the findToFile method can be used.

Statistics retrieval examples using the findToFile method

The <fullClassName> element specifies the statistic type for the XML request to retrieve statistics. The equipment.InterfaceAdditionalStats statistic is the object shown in Code 13-2 and Code 13-3.

Code 13-2 shows a sample request to export performance statistics to an XML file using the findToFile method.

Code 13-2: Sample request to export performance statistics to an XML file

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <security>
        <user>user name</user>
        <password>MD5-hashed password</password>
      </security>
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>

  <SOAP:Body>
    <findToFile xmlns="xmlapi_1.0">
      <fullClassName>equipment.InterfaceAdditionalStats</fullClassName>
    >

    <filter>
    </filter>

    <fileName>ExportStatsToXmlFile_dump.xml</fileName>
    <resultFilter>
```

```
<children/>
    </resultFilter>
  </findToFile>
</SOAP:Body>
</SOAP:Envelope>
```

Code 13-3 shows a sample request to export a filtered set of performance statistics using the `<find>` method.

Code 13-3: Sample request to export performance statistics using the XML API and a filter

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <security>
        <user>user name</user>
        <password>MD5-hashed password</password>
      </security>
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <find xmlns="xmlapi_1.0">
      <fullClassName>equipment.InterfaceAdditionalStats</fullClassName>
      <filter>
        <or>
          <equal name="monitoredObjectSiteId" value="10.1.202.95"/>
          <equal name="monitoredObjectSiteId" value="10.1.202.94"/>
        </or>
      </filter>
    </find>
  </SOAP:Body>
</SOAP:Envelope>
```

Statistics recovery after OSS connection loss

If an OSS loses connectivity to the 5620 SAM server and statistics records are not being retrieved, the OSS client needs to record the time of the most recent statistics retrieval. When the connectivity is restored, the OSS client must send specific `findToFile` requests for the missed statistics data. The OSS must be aware of the different Statistics Policy retention times so that it does not request older log records that have been purged from the 5620 SAM database.

Missing performance statistics

It is possible that in between 5620 SAM performance statistics collection intervals, there are no statistics detected by the OSS. This means that performance statistics log records are missing from the 5620 SAM database. If these statistics records are missing, the 5620 SAM server does not attempt to retry collection from the NEs. Therefore, an OSS client does not need to attempt to retrieve the missing statistics log records for the collection interval, and can continue to process statistics for the next collection interval.

XML statistics output file

Statistics retrieved from the 5620 SAM database in an XML format contain a set of elements and attributes that define the statistic class.

Code 13-4 shows an extract from the XML output file for the statistic counter receivedTotalOctets (ifHCInOctets from IF-MIB) in the equipment.InterfaceAdditionalStats statistic object.

Code 13-4: XML output example for statistics

```
<findToFileResponse xmlapi_1.0">
  <equipment.InterfaceAdditionalStatsLogRecord>
    <monitoredObjectClass>equipment.PhysicalPort</monitoredObjectClass>
    <monitoredObjectPointer>network:10.1.202.93:shelf-1:cardSlot-1:card:daughterCardSlot-
1:daughterCard:port-3</monitoredObjectPointer><displayName>Port
1/1/3</displayName><monitoredObjectSiteId>10.1.202.93</monitoredObjectSiteId>
    <monitoredObjectSiteName>sim202_93</monitoredObjectSiteName>
    <timeCaptured>1127878285113</timeCaptured>
    <periodicTime>938610</periodicTime>
  <suspect>false</suspect>
  <objectFullName>equipment.InterfaceAdditionalStatsLogRecord.equipment.InterfaceAdditi
onalStats30-346</objectFullName>
  <name>equipment.InterfaceAdditionalStats30-346</name>
  <createdOnPollType>ScheduledFullNodeResync</createdOnPollType>
  <updatedOnPollType>ScheduledFullNodeResync</updatedOnPollType>
  <recordId>346</recordId>
  <bucketId>30</bucketId>
  <deploymentState>0</deploymentState>
  <receivedTotalOctets>0</receivedTotalOctets>
  <receivedTotalOctetsPeriodic>0</receivedTotalOctetsPeriodic>
  <receivedUnicastPackets>0</receivedUnicastPackets>
  <receivedUnicastPacketsPeriodic>0</receivedUnicastPacketsPeriodic>
  <receivedMulticastPackets>0</receivedMulticastPackets>
  <receivedMulticastPacketsPeriodic>0</receivedMulticastPacketsPeriodic>
  <receivedBroadcastPackets>0</receivedBroadcastPackets>
  <receivedBroadcastPacketsPeriodic>0</receivedBroadcastPacketsPeriodic>
  <transmittedTotalOctets>0</transmittedTotalOctets>
  <transmittedTotalOctetsPeriodic>0</transmittedTotalOctetsPeriodic>
  <transmittedUnicastPackets>0</transmittedUnicastPackets>
  <transmittedUnicastPacketsPeriodic>0</transmittedUnicastPacketsPeriodic>
  <transmittedMulticastPackets>0</transmittedMulticastPackets>
  <transmittedMulticastPacketsPeriodic>0</transmittedMulticastPacketsPeriodic>
  <transmittedBroadcastPackets>0</transmittedBroadcastPackets>
  <transmittedBroadcastPacketsPeriodic>0</transmittedBroadcastPacketsPeriodic>
  <children-Set></children-Set>
</equipment.InterfaceAdditionalStatsLogRecord>
</findToFileResponse>
```

Table 13-2 provides a description of the XML elements and attributes shown in Code 13-4.

Table 13-2 Description of the statistic XML element attributes

Element	Attribute	Description
<class>	name	Package and class name for the statistic class being retrieved from the database
	xmlTag	Alias of the class name if a conflict exists between two class names
	displayed	Name of the class displayed on the 5620 SAM GUI
	cli	CLI command to view the statistics on the node

(1 of 2)

Element	Attribute	Description
<property>	name	Name of the statistic counter
	type	Type of statistic
	default	Default value of the statistic counter, if applicable
	loggableType	Method for storing the counter on the node. The counter can be stored as raw data, an absolute counter, or as a rate. In the case of an absolute counter, a Periodic Statistic Counter is also stored on the 5620 SAM and calculates the delta between the values retrieved for the current and previous intervals
	snmpCounter	MIB information for the counter
	dependsOn	Parent counter for the statistic. Applicable for periodic statistics
<Description>	—	Description of the statistic counter function

(2 of 2)

13.10 Statistics monitoring using JMS

The 5620 SAM supports the publishing of polled statistics measurements to OSS clients using JMS on the XML statistics topic.



Note — This section describes a deprecated method of publishing polled statistics measurements to OSS clients and is not recommended by Alcatel-Lucent.

If the configuration of the topics specifies statistics on the same topic as other message types, you can filter the client statistics and accounting data using the filter on a subscription. The filter is based on the category of the message, for example, statistics or accounting data. The default setting for the 5620 SAM is to send statistics and accounting messages to all clients.

A client can stop listening to the messages by setting one or more filters on a subscription identifying that the category of message should not include statistics or accounting data. A maximum of ten 5620 SAM-O clients can listen to the messages.

See section [4.3](#) for more information about subscribing to JMS topics.

JMS XML statistics notifications are generated whenever scheduled statistics of the corresponding statistics policy are modified. You can disable this function by setting the Events Enabled parameter in the statistics policy to false.

13.11 Collecting JMS performance statistics

You can use the 5620 SAM to collect the following JMS performance statistics:

- JMS Subscriber Topic
- JMS Subscriber Session
- Publisher XML Event

Real-time collection of JMS statistics is not supported because of the potential performance effect on a 5620 SAM main server.

See the *5620 SAM Statistics Management Guide* for information about the JMS statistics counters.

13.12 Sample request to retrieve historical statistics from an NE

Code 13-5 shows a sample request to retrieve historical statistics from an NE using the `findToFile` method. An exception-free response indicates that the 5620 SAM-O received and validated the request.

Code 13-5: Sample request to retrieve historical statistics from an NE

```
<SOAP:Body>
  <findToFile xmlns="xmlapi_1.0">
    <fullClassName>equipment.InterfaceAdditionalStatsLogRecord</fullClassName>
    <filter>
      <and>
        <equal name="monitoredObjectPointer"
value="network:10.1.202.93:shelf-1:cardSlot-1:card:daughterCardSlot-1:daughterCard:port-3"/>
        <between name="timeCaptured" first="1127142900000"
second="1127143800000"/>
      </and>
    </filter>
    <fileName>Equipment.InterfaceAdditionalStatsLogRecord.xml</fileName>
  </findToFile>
</SOAP:Body>
```

13.13 Sample request to collect interface statistics

Code 13-6 shows a sample request to perform the following tasks using the `findToFile` method:

- collect interface statistics
- format the collected statistics in a XML file for post-processing

To return the data using the XML API, you can create a similar set of filters and use the `<find>` method.

The following conditions apply to the sample shown in Code 13-6:

- uses the `findToFile` method, which can use any class from any other package
- specifies the `<propertyFormat>`, which indicates the number of parameters (elements) to retrieve, either `all` or `configOnly`
- exports the collected data to an XML file, with a filename of `stats_file.xml`

When you create export files using the `findToFile` method, you should be aware of the following:

- the export files are placed in a directory specified during installation
- no FTP server is installed during the 5620 SAM database installation, so to export files from the specified directory there can be an FTP server installed on the machine where the database is installed, if necessary. If these files are located on a remote disk, FTP may not be necessary to access them.

An exception-free response indicates that the 5620 SAM-O received and validated the request.

Code 13-6: Sample request to collect interface statistics

```
<SOAP:Body>
  <findToFile xmlns="xmlapi_1.0">
    <fullClassName>equipment.InterfaceAdditionalStatsLogRecord</fullClassName>
    <filter>
      <and>
        <equal name="monitoredObjectPointer"
value="network:10.1.202.93:shelf-1:cardSlot-1:card:daughterCardSlot-1:daughterCard:port-3"/>
        <between name="timeCaptured" first="1127142900000"
second="1127143800000"/>
      </and>
    </filter>
    <fileName>Equipment.InterfaceAdditionalStatsLogRecord.xml</fileName>
  </findToFile>
</SOAP:Body>
```

Configuration management

- 14 – Configuration management overview**
- 15 – Device configuration management**
- 16 – Network configuration management**
- 17 – Policy configuration management**
- 18 – Service configuration management**
- 19 – Script and template configuration management**

14 – Configuration management overview

- 14.1 Configuration management overview 14-2**
- 14.2 Configuration methods 14-2**
- 14.3 Deployers 14-7**
- 14.4 Configuration management guidelines for multiple requests 14-15**
- 14.5 Workflow to handle deployer failures 14-16**

14.1 Configuration management overview

OSS applications can use the 5620 SAM-O XML interface to perform configuration operations on 5620 SAM-managed network objects and their associated child objects. This is achieved by sending XML requests that allow the OSS to create, delete, or modify existing objects. These requests can also extend to the object's children at the same time, if required.



Note — The configuration operations that are allowed on each object is defined in the XML schema. See the *5620 SAM-O XML Schema*. See the *5620 SAM-O XML Reference* for more information about the object hierarchies (relationship between parent and child objects).

The OSS user accounts privileges determine the access and restrictions for performing configuration operations on the various 5620 SAM functional areas. This needs to be configured appropriately so as to give the OSS user access to a particular functional area of the 5620 SAM. See the user security section of the *5620 SAM User Guide* for more information about configuring user accounts.

Most of the configuration operations that can be performed on the functional areas of the 5620 SAM using the GUI can also be performed using the 5620 SAM-O interface. The main areas for OSS configuration management include:

- device configuration—for example, routers, cards, ports, GNE, and channels. See chapter 15 for more information.
- network configuration—for example, protocols, MPLS, service tunnels, and VRRP. See chapter 16 for more information.
- policy configuration—for example, service policy, routing policy, and network policy. See chapter 17 for more information.
- service configuration—for example, VLL, VPLS, VPRN, VLAN, mirror composite, and customer/subscriber. See chapter 18 for more information.
- scripts and template configuration—See chapter 19 for more information.

14.2 Configuration methods

The 5620 SAM-O uses the generic package to define methods that are applicable to all objects. Methods for object configuration (create, modify, delete) are provided by the generic.GenericObject class. The most commonly used configuration methods from this class are described in Table 14-1.

Table 14-1 Object configuration methods using the generic.GenericObject class

Method	Description
generic.GenericObject.configureInstance	Modifies an existing object and associated children. Not recommended for OSS clients as there are no return parameters.

(1 of 2)

Method	Description
<code>generic.GenericObject.configureInstanceWithResult</code>	Modifies an existing object and associated children. The method returns the newly configured parameters on the object and child, including parameters set due to associated conditions. See the <i>5620 SAM-O XML Reference</i> for input parameters. See Code 14-2 for an example of the method.
<code>generic.GenericObject.configureChildInstance</code>	Creates a child of the object with the desired configuration values for the child or children objects, and optional grandchildren. See the <i>5620 SAM-O XML Reference</i> for input parameters.
<code>generic.GenericObject.configureChildInstanceWithResult</code>	Creates a child of an existing object. The method returns the newly configured parameters on the child object and grandchildren, including parameters set due to associated conditions. See the <i>5620 SAM-O XML Reference</i> for input parameters. See Code 14-3 for an example of the method.
<code>generic.GenericObject.deleteInstance</code>	Delete an existing object. See Code 14-6 for an example of the method.

(2 of 2)

The following section provides XML sample requests to do the following configuration operations:

- configure an existing port
- create an epipe service
- delete an epipe service

These samples are included in the Basic SDK. See Appendix B for more information about the full list of Basic SDK sample requests and associated responses.



Caution — The messages in this section are examples of the SOAP XML request format. Use the sample as a base to build your request. Ensure that you test your request before network deployment.

To compose a configuration command that modifies, creates, or deletes an existing object, you need the following information:

- the appropriate generic configuration method
- the class of the object you want to configure (not required if deleting)
- the full name of the object you want to configure
- the deployer type (typically immediate). See section 14.3 for more information about deployers.

Modify example: To configure an existing port

The following example outlines the parameters required for an XML request to set the description on a port.

To compose a modification command, the key input parameters include:

- method - generic.GenericObject.configureInstanceWithResult
- class - equipment.PhysicalPort
- full name -
network:35.121.20.40:shelf-1:cardSlot-1:card:daughterCardSlot-1:daughterCard:port-10

Code 14-1 shows a sample request to set the description for a port (ConfigureObject.xml).

Code 14-1: Sample request to set the description for a port (ConfigureObject.xml)

```
<SOAP:Body>
  <generic.GenericObject.configureInstanceWithResult
xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <distinguishedName>network:35.121.20.40:shelf-1:cardSlot-1:card:
daughterCardSlot-1:daughterCard:port-10</distinguishedName>
    <includeChildren>false</includeChildren>
    <configInfo>
      <equipment.PhysicalPort>
        <actionMask><bit>modify</bit></actionMask>
        <description>to Mumbai</description>
      </equipment.PhysicalPort>
    </configInfo>
  </generic.GenericObject.configureInstanceWithResult>
</SOAP:Body>
<SOAP:Envelope>
```

For this sample request, other key input parameters include:

- includeChildren - set to false, means do not include child info in the results
- configInfo - desired values to which to configure object, and optionally children
 - actionMask - specifies the modify operation
 - description - specifies the attribute and value of the equipment.PhysicalPort class that is being modified

The results of the command are shown in Code 14-2. Results include all of the port attributes, but no children (as per includeChildren) in the configInfo parameter.

Code 14-2: Sample response to set the description for a port

```
<generic.GenericObject.configureInstanceWithResultResponse
xmlns="xmlapi_1.0">
  <configInfo>
    <equipment.PhysicalPort>
      <usedAsSyncReference>none</usedAsSyncReference>
      <cardSlotId>1</cardSlotId>
      <daughterCardSlotId>1</daughterCardSlotId>
      <specificCardType>
        <bit>mda_m60_faste_tx</bit>
      </specificCardType>
      <description>to Mumbai</description>
      [...]
      <selfAlarmed>true</selfAlarmed>
      <children-Set>
    </equipment.PhysicalPort>
  </configInfo>
</generic.GenericObject.configureInstanceWithResultResponse>
```

Create example: To create an Epipe object

The following example outlines what is required for an XML request to create an Epipe object. The example only shows object creation, and does not include all of the child objects required to create a working service.

To compose a creation command, the key input parameters include:

- method - generic.GenericObject.configureChildInstanceWithResult
- class - epipe.Epipe
- parent - svc-mgr. The fully distinguished name of the parent can be determined by looking at the Parent Hierarchy of the class to be created in the *5620 SAM-O XML Reference*.

Code 14-3 shows a sample request to a create an Epipe object.

Code 14-3: Sample request to create an Epipe object (CreateObject.xml)

```
<SOAP:Body>
  <generic.GenericObject.configureChildInstanceWithResult
    xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <synchronousDeploy>true</synchronousDeploy>
    <distinguishedName>svc-mgr</distinguishedName>
    <childConfigInfo>
      <epipe.Epipe>
        <actionMask><bit>create</bit></actionMask>
        <displayName>Epipe Example 1</displayName>
        <description>Epipe Example 1</description>
        <subscriberPointer>subscriber:1</subscriberPointer>
      </epipe.Epipe>
    </childConfigInfo>
  </generic.GenericObject.configureChildInstanceWithResult>
</SOAP:Body>
<SOAP:Envelope>
```

For this sample request, other key input parameters include:

- distinguishedName - set to svc-mgr, this is the pointer to the parent object
- childConfigInfo - desired values to configure the children information, in this case the child object is epipe.Epipe
 - actionMask - set to create, specifies the creation operation
 - displayName, description, and subscriberPointer - these are the attributes and associated values that are set on the epipe.Epipe object

The results of the command are shown in Code 14-4. Results include all of the default epipe attributes including the displayName, description, and subscriberPointer that were set in the request.

Code 14-4: Sample response to creating an epipe

```
<generic.GenericObject.configureChildInstanceWithResultResponse
  xmlns="xmlapi_1.0">
  <childConfigInfo>
    <epipe.Epipe>
      <lastCacTime>0</lastCacTime>
      <cacStatus>notApplicable</cacStatus>
      <cacProbableCause>notApplicable</cacProbableCause>
      <defaultVcId>49</defaultVcId>
      <topologyAutoCompletion>false</topologyAutoCompletion>
      <transportPreference>any</transportPreference>
    </epipe.Epipe>
  </childConfigInfo>
</generic.GenericObject.configureChildInstanceWithResultResponse>
```

```

        <useBwReservedPath>noPreference</useBwReservedPath>
        [...]
        <objectFullName>svc-mgr:service-10907</objectFullName>
        <displayName>Epipe Example 1</displayName>
        <description>Epipe Example 1</description>
        <subscriberPointer>subscriber:1</subscriberPointer>
        <selfAlarmed>false</selfAlarmed>
        <children-Set>
        </epipe.Epipe>
    </childConfigInfo>
</generic.GenericObject.configureChildInstanceWithResultResponse>

```



Note — If you wanted to only return some of the attributes of the Epipe, this can be done by including a `<resultFilter>` tag after the `childConfigInfo` in the request. The sample in Code 14-5 will return the fullname of the Epipe service name.

Code 14-5: Sample `<resultFilter>`

```

<resultFilter>
  <attribute>objectFullName</attribute>
</resultFilter>

```

The full code sample that includes the `resultFilter` is included in the Basic SDK - `CreateObjectWithResultFilter.xml`. See section 12.3 for more information about creating filters.

Delete example: To delete an Epipe object

The following example outlines what is required for an XML request to delete an Epipe service.



Caution — Deleting an object deletes all of its child objects, so deleting the wrong object could delete an entire tree of objects.

To compose a deletion command, the key input parameters include:

- method - `generic.GenericObject.deleteInstance`
- full name - `svc-mgr:service-10907` (created by the Create example in Code 14-3)

Code 14-6 shows a sample request to delete an Epipe object.

Code 14-6: Sample request to delete an Epipe object (DeletObject.xml)

```

<SOAP:Body>
  <generic.GenericObject.deleteInstance xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <synchronousDeploy>true</synchronousDeploy>
    <distinguishedName>svc-mgr:service-10907</distinguishedName>
  </generic.GenericObject.deleteInstance>
</SOAP:Body>
<SOAP:Envelope>

```

The results of the command is in Code 14-7. An exception-free response indicates that the 5620 SAM-O received an validated the request.

Code 14-7: Sample response to deleting an epipe

```
<generic.GenericObject.deleteInstanceResponse xmlns="xmlapi_1.0"/>
```

14.3 Deployers

Deployers model the communication requests made from the management domain via the 5620 SAM GUI or OSS to the network. Deployers are created by the 5620 SAM at system startup and are present in a finite deployer pool to be used to:

- perform actions from the 5620 SAM to the managed network
- queue configuration requests from both client GUIs and OSSs

Deployer objects have data about the current state of deployment and can be used to track changed objects and parameters (elements) that need to be sent to the network. When the deployer performs the network change, all data is cleared from the deployer. If the deployer fails, it attempts to redeploy the request based on the deployer policy configuration.

OSS applications that send configuration requests to the network must monitor deployers. When changes are sent to the network, deployers are created, queued, and dispatched to the managed routers. When the deployment is successful, the 5620 SAM notifies the OSS that the deployment is completed for the request that triggered the deployment. If the deployment fails, an alarm creation or change event is generated and sent using JMS. It is important that OSSs clear deployers after they are used.

See the following sections for more information:

- [“Deployer failures”](#) in this section for more information about managing deployer failures
- [“Deployer failure recovery procedures”](#) in this section for more information about how to recover from deployer failure, depending on the type of configuration request
- [“Deployer alarms and events”](#) in this section for more information about the deployer alarm format and DeployerEvent
- [chapter 4](#) for more information about JMS events

Deployment types <deployer>

Deployment related methods are found in the generic package and all configuration methods require a specified deployer type, as specified in the <deployer> tag. The deployer type is specified in the xmlApiTypes.xsd file. Deployers specify how and when deployment occurs. The valid values (with numeric equivalents) are:

- immediate (-1) to deploy to the network immediately
- followTransaction (-2) reserved for future use
- postponed (-3) reserved for future use
- ignored (-4) to not deploy to the network, so changes are only added to the database

Synchronous and asynchronous requests

There are two types of deployments you can configure:

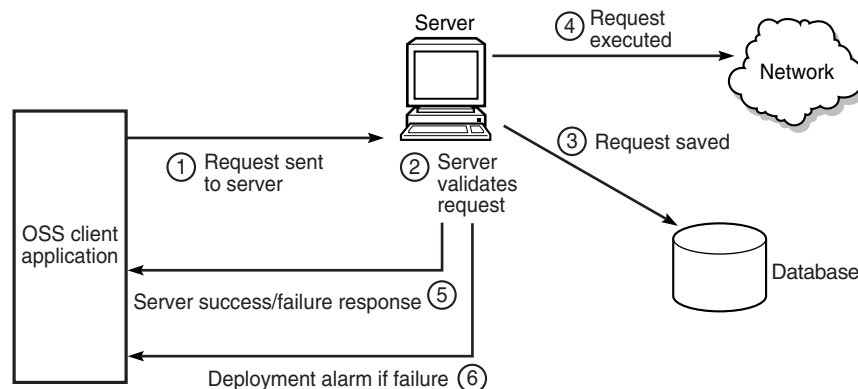
- asynchronous (default)
- synchronous

Synchronous requests

Users can specify synchronous network requests using the `<synchronousDeploy>` and other tags. These requests are sent by the OSS to the 5620 SAM server where the request is validated and then saved to the 5620 SAM database. Subsequently, a deployer is created and queued on the 5620 SAM server for dispatching to the network. When the deployer has successfully performed the deployment across the network, or when all of the retries specified in the request are completed, the synchronous response is returned and the deployer put back in the pool of available deployers.

If there is a failure of all retries, an alarm is raised that contains details of the failed deployer and a failure XML response is sent back containing the failed deployer Id. See [“Deployer alarms and events”](#) in this section for more information. Figure 14-1 shows a synchronous network deployment.

Figure 14-1 Database interaction with synchronous network deployment



17752

Alcatel-Lucent recommends the following strategy to minimize the use of resources and the blocking of synchronous OSS requests:

- Set the `<clearOnDeployFailure>` parameter to true to automatically clear deployers if a failure occurs. Default=false.
- Do not change the default value (0) of the `<deployRetries>` and `<deployRetryInterval>` parameters.

Code 14-8 shows the setting of previously discussed deployment parameters when deleting an object. The parameters `<deployRetries>` and `<deployRetryInterval>` may be omitted to use default values.

Code 14-8: Sample request for synchronous deployment

```

<SOAP:Body>
  <generic.GenericObject.deletedInstance xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <synchronousDeploy>true</synchronousDeploy>
    <clearOnDeployFailure>true</clearOnDeployFailure>
    <deployRetries>0</deployRetries>
    <deployRetryInterval>0</deployRetryInterval>
    <distinguishedName>network:10.1.202.93</distinguishedName>
  </generic.GenericObject.deleteInstance>
</SOAP:Body>

```

Asynchronous requests

Asynchronous requests are sent by the OSS to the 5620 SAM server where the request is validated and then saved to the 5620 SAM database. Subsequently, a response from the server is sent back to the OSS. A deployer is created and queued on the 5620 SAM server for dispatching to the network. When the deployer has successfully performed the deployment across the network, or when all of the retries specified in the request are completed, the deployer is put back in the pool of available deployers.

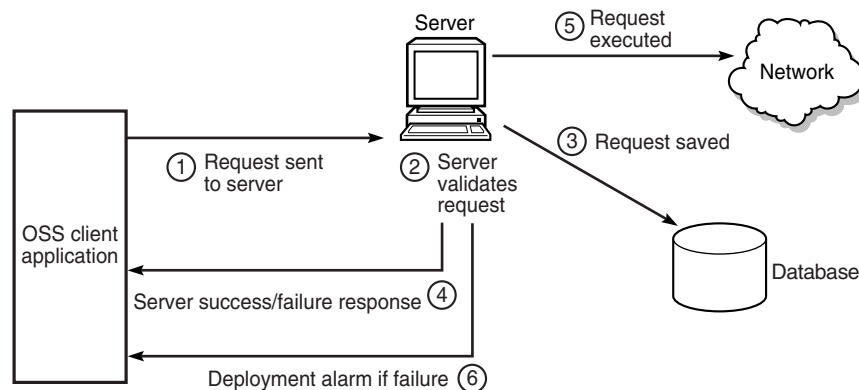
If there is a failure of all retries, an alarm is raised that contains details of the deployer. See [“Deployer alarms and events”](#) in this section for more information. The default setting is that requests are made asynchronously across the network.



Note — If the OSS is not listening for alarms, it will not know if the deployments was successful, and may have to retrieve deployment information to verify. See [“Deployer failures”](#) in this section for more information. Alcatel-Lucent recommends implementing JMS to listen for deployment alarms or the DeployerEvent.

Figure 14-2 shows an asynchronous network deployment.

Figure 14-2 Database interaction with asynchronous network deployment



17288

The sample synchronous request shown in Code 14-8 can be changed to an asynchronous request by changing `<synchronousDeploy>` to `false`:

```

<synchronousDeploy>false</synchronousDeploy>

```

Deployer failures

In the case of synchronous request failures, the 5620 SAM-O notifies the OSS of the deployer failure in the returned XML response. For asynchronous request failures, the OSS will not receive an XML response about the deployment failure. For both cases, the OSS must subsequently perform procedures for error-recovery.

If the OSS connection is lost or times out before the 5620 SAM-O returns a result to the OSS application, the OSS may not be aware of any subsequent success or failure notifications of the request. Steps must be taken at a later time, either manually or through the OSS, to verify whether the request was successful. If the request was unsuccessful, for example, because of a failed deployer, the OSS must perform procedures for error-recovery.



Note 1 – It is recommended that the OSS application record configuration failures for future reference in error recovery or troubleshooting.

Note 2 – If failures occur, deployers are left in an undeployed state and must be cleared. Uncleared deployments consume resources on the 5620 SAM server and can block further synchronous OSS requests.

Alcatel-Lucent recommends the following strategy to minimize the use of resources and the blocking of synchronous OSS requests:

- Set the `<clearOnDeployFailure>` parameter to true to automatically clear deployers if a failure occurs.
- Do not change the default value (0) of the `<deployRetries>` and `<deployRetryInterval>` parameters.

Contact your Alcatel-Lucent OIPS technical support representative for information about how to optimize deployer performance so that resource consumption and the blocking of synchronous OSS requests to network elements is minimized.

Error recovery methods

If there is a deployment failure, the failed deployer has more information on the failed state of the deployment for the OSS to retrieve and recover. Table 14-2 describes the methods from the generic.GenericObject class that are available to help with recovering from failed deployers.

Table 14-2 Generic methods for error recovery

Generic method	Description
<code>generic.GenericObject.getDeployers</code>	To find information about deployers that matches a specified filter with the children hierarchy. For example, you can get the deployer IDs for deployers with the failed deployment states set in the <code>generic.GenericObject.deploymentState</code> parameter. See <code>Score.DeploymentState</code> in the <i>5620 SAM-O XML Reference</i> to see all the valid failure values.
<code>generic.GenericObject.getDeployer</code>	To find all parameter values for a specific deployer ID

(1 of 2)

Generic method	Description
<code>generic.GenericObject.clearDeployer</code>	To clear the deployer with the specified deployer ID
<code>generic.GenericObject.triggerResync</code>	To perform a non-scheduled resynchronization of an object
<code>generic.GenericObject.getDeployersShallow</code>	To find matched deployers similar to <code>getDeployers</code> without children hierarchy

(2 of 2)

Filtering can be applied when retrieving information about deployers using the following methods:

- `generic.GenericObject.getDeployers`
- `generic.GenericObject.getDeployer`
- `generic.GenericObject.getDeployersShallow`

Code 14-9 shows an example of filtering on the `generic.DeployerInfo` class returning all deployers fullnames where the deployment state failed due to an internal error (16).

Code 14-9: Sample of defining the filter class on the `generic.DeployerInfo`

```
<filter>
  <and class="generic.DeployerInfo">
    <equal name="state" value="16" />
    <wildcard value="Default.DeployerBank:depl-%" name="objectFullName" />
  </and>
</filter>
```

Code 14-10 shows a failed response (exception) for a deployer request. When all retries fail, the failure response indicates the failed deployer ID. An alarm is not raised until all retries are attempted. See Appendix B for information about how to obtain a sample request to retrieve deployer information.

Code 14-10: Sample failed response for a deployer request

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP:Header>
  <header xmlns="xmlapi_1.0">
    <requestID>clientName@requestId</requestID>
    <requestTime>11-Dec-2006 10:53:48 AM</requestTime>
    <responseTime>11-Dec-2006 10:53:49 AM</responseTime>
  </header>
</SOAP:Header>

<SOAP:Body>
  <generic.GenericObject.configureChildInstanceException
xmlns="xmlapi_1.0">
    <description>
      [ app: generic ] [ class: generic.GenericObject ] [ instance: N/A ] [ descr:
      Operation Failed During Deployment ]
    </description>
    <deployers>
      <deployer>Default.DeployerBank:depl-482</deployer>
      <deployer>Default.DeployerBank:depl-480</deployer>
    </deployers>
  </generic.GenericObject.configureChildInstanceException>
</SOAP:Body>
```

</SOAP:Envelope>



Note — The <responseTime> tag in a 5620 SAM-O header is the time at which the response stream is opened.

Deployer failure recovery procedures

The deployer failure recovery procedures to be carried out varies with the type of action (creation, modification, or deletion):

- If deployment fails during object creation, you must clear the deployers to remove the previously created object from the database. To remove the possibility of partially-created objects on the node, resynchronize the object (this can be done using the `generic.GenericObject.triggerResync` method) and remove all partially-created objects using the 5620 SAM-O or the 5620 SAM.
- If deployment fails during object modification, clearing a deployment does not reset the modified object in the database. To reset to the previous object setting, resynchronize the object. In there are partially-modified objects, set the objects back to their original values using the 5620 SAM-O or the 5620 SAM.
- If deployment fails during object deletion, clearing a deployment completes the object deletion in the 5620 SAM database. To reset to the previous object setting, resynchronize the object.

See [“Workflow to handle deployer failures”](#) in this chapter for more information about handling a deployment error that has caused the 5620 SAM database to be out of synchronization with the managed network.

Deployer alarms and events

OSS applications that choose to rely on alarms and events to notify them of deployer failures need to subscribe to the JMS interface. See section [4.6](#) for more information.

The `DeployerEvent` indicates the success or failure of an asynchronous deployment request and will return a `successList` and `failedList` of `deployerIds`. See [Code 4-8](#) for a sample `DeployerEvent`. An OSS can then take the failed `deployerId` and retrieve more information on the reason for the failure and then perform the required recovery procedure.

Deployment alarms are also raised and OSS applications can use the `requestId` and the `requestUser` to match failures with specific requests. After the `deployerId` is extracted from the alarm, you can use the `deployerId` to query a specific deployer. See [Appendix B](#) for information about how to obtain a sample request to retrieve deployer information.

[Code 14-11](#) shows a sample deployment alarm. The following information is contained in the deployer alarm:

- the `alarmClass` attribute identifies the alarm class as a deployment failure
- the deployer ID is last component of the `objectName` attribute, in this case `deployerId=771`

- the additionalText attribute also contains the deployerId, the requestId, and the client user (requestUser) that made the request in the format:
 deployerId=771;requestId=AreqGenericObjectConfigureInstance-CLIENT-admin-5620SAM@138.120.135.164-15;requestUser=user
 name;deploymentType=3, where *user name* is the account name of the operator account in the SOAP request.

Code 14-11: Sample deployment alarm

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <fm.FaultManager.findFaultsResponse xmlns="xmlapi_1.0">
      <result>
        <fm.AlarmInfo>
          <severity>minor</severity>
          <previousSeverity>indeterminate</previousSeverity>
          <originalSeverity>minor</originalSeverity>
          <highestSeverity>minor</highestSeverity>
          <probableCause>11</probableCause>
          <alarmName>13</alarmName>
          <type>5</type>
          <affectedObjectFullName>network:10.1.202.93:shelf-1:cardSlot-
1:card:daughterCardSlot-1:daughterCard:port-5</affectedObjectFullName>
          <affectedObjectClassName>equipment.PhysicalPort</affectedObjec
ctClassName>
          <isAcknowledged>>false</isAcknowledged>
          <wasAcknowledged>>false</wasAcknowledged>
          <acknowledgeBy>N/A</acknowledgeBy>
          <firstTimeDetected>1128369209032</firstTimeDetected>
          <lastTimeDetected>1128369209032</lastTimeDetected>
          <lastTimeSeverityChanged>0</lastTimeSeverityChanged>
          <lastTimeCleared>0</lastTimeCleared>
          <lastTimePromoted>0</lastTimePromoted>
          <lastTimeDemoted>0</lastTimeDemoted>
          <lastTimeEscalated>0</lastTimeEscalated>
          <lastTimeDeEscalated>0</lastTimeDeEscalated>
          <lastTimeAcknowledged>0</lastTimeAcknowledged>
          <frequency>0</frequency>
          <occurrences>N/A</occurrences>
          <numberOfOccurrences>2</numberOfOccurrences>
          <numberOfOccurrencesSinceClear>2</numberOfOccurrencesSinceClear
>
          <numberOfOccurrencesSinceAck>0</numberOfOccurrencesSinceAck>
          <isServiceAffecting>>false</isServiceAffecting>
          <additionalText>deployerId=771;requestId=AreqGenericObjectCon
figureInstance-CLIENT-admin-5620SAM@13@138.120.135.164-15;requestUser=admin;
deploymentType=3;</additionalText>
          <operatorAssignedUrgency>indeterminate</operatorAssignedUrgen
cy>
          <urgencyAssignedBy>1</urgencyAssignedBy>
          <relatedObjects>
            <null/>
          </relatedObjects>
          <affectingObjects>
            <relationshipTreeList>
              <relationshipTree>
                <objectFullName>Network
Queue:default</objectFullName>
                <subtree/>
              </relationshipTree>
            </relationshipTreeList>
          </affectingObjects>
        </fm.AlarmInfo>
      </result>
    </fm.FaultManager.findFaultsResponse>
  </SOAP:Body>
</SOAP:Envelope>
```

```

        </affectingObjects>
        <nodeId>10.1.202.93</nodeId>
        <nodeName>sim202_93</nodeName>
        <affectedExceptionDisplayedName>Port
1/1/5</affectedExceptionDisplayedName>
        <applicationDomain>equipment</applicationDomain>
        <displayedClass>PhysicalPort</displayedClass>
        <alarmClassTag>generic.DeploymentFailure</alarmClassTag>
        <affectedExceptionClassIndex>87</affectedExceptionClassIndex>
        <affectedExceptionInstanceIndex>5</affectedExceptionInstanceIndex>
        <deployerName>N/A</deployerName>
        <deployerId>0</deployerId>
        <requestId>N/A</requestId>
        <requestUser>N/A</requestUser>
        <objectFullName>faultManager:network@10.1.202.93@shelf-1@card
Slot-1@card@daughterCardSlot-1@daughterCard@port-5|alarm-13-5-11-deployerId=
771</objectFullName>
        <objectClassName>fm.AlarmObject</objectClassName>
        <allomorphicClassName>fm.AlarmObject</allomorphicClassName>
        <objectId>-578593836</objectId>
        <displayName>Port 1/1/5 -
network@10.1.202.93@shelf-1@cardSlot-1@card@daughterCardSlot-1@daughterCard@
port-5|alarm-13-5-11-deployerId=771</displayName>
        <lifeCycleState>1</lifeCycleState>
        <deploymentState>0</deploymentState>
        <neId>10.1.202.93</neId>
    </fm.AlarmInfo>
</result>
</fm.FaultManager.findFaultsResponse>
</SOAP:Body>
</SOAP:Envelope>

```

Deployer simulation

By default, the 5620 SAM server deploys configuration changes to network elements through SNMP. You can configure the 5620 SAM-O to save configuration changes in the 5620 SAM database without deploying the changes, which facilitates OSS development in the absence of real equipment or simulators.

You can use the 5620 SAM-O to configure a mediation.DeploymentPolicy with a mediation.DeploymentMode property set to 1, to disable deployment. The default setting is 2, SNMP.

See chapter 17 for more information about how to configure policies.



Caution 1 — Since configuration errors may not always be caught by the 5620 SAM server before configuration changes are committed to the database, you must disable deployer simulation and test with live network equipment to validate changes prior to OSS deployment in the production network.

Caution 2 — Deployment should never be disabled in a production network.

14.4 Configuration management guidelines for multiple requests

Multiple requests can be sent in one SOAP body using streamed HTTP connections. The successful or failed results are streamed back to the OSS application in the order processed. For synchronous requests processing stops for the SOAP body when the first failed request is encountered or when all requests are successful. If a failure occurs, subsequent requests in the same multi-part request are not processed.

The time required to complete each deployment request depends on the SNMP deployment parameters and complexity of the request. If a network element is not reachable through SNMP, it may take much longer for the request to complete.

If you need to send consecutive XML API requests that affect interdependent objects, Alcatel-Lucent recommends that you allow each request to fully complete before proceeding with a subsequent configuration request. Examples of interdependencies are as follows:

- When a subsequent configuration request is dependent on a previous configuration request. For example, physical ports cannot be configured until the card has been configured.
- When a configuration request on one object causes side-effect changes on the same or another object that you subsequently need to configure.

The wait-time duration between requests depends on several factors, including:

- network latency
- SNMP trap incoming rate on the 5620 SAM server
- processing load on the 5620 SAM server and the network devices
- potential SNMP trap loss rate

See the *5620 SAM-O XML Reference* for information about object relationships. Alcatel-Lucent recommends that you configure and delete network objects one object at a time to facilitate troubleshooting of potential problems, such as deployer failures.

14.5 Workflow to handle deployer failures

The following workflow outlines the high-level steps necessary to handle a deployment error that may cause the 5620 SAM database to be out of synchronization. Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the Advanced SDK-Misc package for the steps below.

- 1 Clear the deployer. The XML configuration request uses the `GenericObject` method `clearDeployer` on the specified `deployerName` which identifies the deployer to be cleared.
- 2 Resynchronize the object that caused the deployment error. The XML configuration request to resynchronize the object using the `generic.GenericObject.triggerResync` method.



Note — When you use the `generic.GenericObject.triggerResync` method for a specific `<instanceName>`, you can specify whether a resynchronization on the object is performed by setting the `<resyncSelf>` flag to true, or whether the resync is performed on its children by setting the `<resyncChildren>` flag to true.

The 5620 SAM does not notify 5620 SAM-O clients when objects are resynchronized.

- 3 Perform one of the following steps, depending on your 5620 SAM-O client implementation.
 - a Set the object back to the previous setting. To reset the previous object setting, resynchronize the object by using the `generic.GenericObject.triggerResync` method described in step 2.

See “[Deployer failure recovery procedures](#)” in section 14.3 for more information about the different request types that you must consider when you set an object back to the previous setting.

For example, if a request to create an object with children objects successfully creates the parent object, but creates only some of the children objects, you should create a request to delete the partially created objects.

For partial failures on a request to modify a port mode, the OSS client should create a request to change the port mode to return it to the original setting.

For failed deletion requests, performing step 2 resynchronizes the 5620 SAM with the network. Depending on the state the deleted object or objects in the 5620 SAM, the 5620 SAM-O client should create a request to re-create the deleted objects to restore the 5620 SAM and the network back to their original state before the failed deployment.



Warning — Subsequent requests to set objects back to their previous settings may fail. After an unsuccessful attempt to set objects back to their previous setting, Alcatel-Lucent recommends that the OSS client not retry.



Note — The 5620 SAM continues to try and deploy the change, even after the failure, according to the deployment policies configured for the router. If an automatic recovery mechanism is implemented by the OSS application, the OSS application should use a recovery mechanism that is consistent with the configured deployer policies. For example, if the policy defines that deployers should redeploy after a failure, the OSS application should use a timer mechanism to wait until the deployment is retried before attempting another recovery method.

- b Raise an alarm for the administrator to investigate the deployment failure.

In many cases, such as a loss of connectivity, deployments that fail will continue to fail if retried. If an OSS client continues to retry deployments after a failure, 5620 SAM resources will be consumed that will be ineffective, ultimately until all deployer resources are consumed. Alcatel-Lucent recommends that you do not continue to retry failed deployment requests. Instead, the OSS client can raise an alarm to alert the administrator that there is a deployment problem.



Note — Contact your Alcatel-Lucent OIPS technical support representative for information about how to manage deployer failures.

15 – Device configuration management

- 15.1 Device configuration overview 15-2**
- 15.2 Workflow to configure equipment 15-2**
- 15.3 Generic NE profiles 15-3**
- 15.4 Generic NE profile parameter configuration 15-3**
- 15.5 Device software upgrades 15-4**

15.1 Device configuration overview

You can use the 5620 SAM-O to create, configure, and manage the devices and children objects that are part of the 5620 SAM network. Equipment, such as routers, are at the top of the hierarchy and their properties and attributes are defined in the XML schema.

The 5620 SAM-O XML interface uses the following packages to manage equipment:

- equipment for physical equipment
- fr for frame relay equipment
- ethernetequipment for Ethernet equipment
- sonetequipment for SONET and SDH equipment
- sonettiming for SONET and SDH timing
- tdmequipment for TDM equipment
- genericne for Generic NE equipment

Most of the configuration actions that can be performed using the 5620 SAM GUI can also be performed using the 5620 SAM-O. Typical configuration actions include configuring cards, daughter cards, ports, channels, and the associated properties.

15.2 Workflow to configure equipment

The following workflow outlines the high-level steps necessary to configure equipment. Each of the steps provides guidance on the type of XML request required to perform a particular configuration action. Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the advanced SDK configuration equipment package for each step.

- 1 Configure each card slot by sending XML create requests to set the card type on the card on the router.
- 2 Configure each daughter card slot by sending XML create requests to set the daughter card type on the daughter card slot on the router.
- 3 Configure all ports by sending XML modify requests to turn up the port, with options to set encap type and mode.
- 4 As required, configure the channels. Prior to configuring SONET channels, card and ports have to exist. Send XML create and modify requests to create channel and set the parameters on the channels.
- 5 As required, configure LAG and MC-LAG by sending XML create and modify requests to create LAG interfaces and modify the parameters.



Note — Alcatel-Lucent recommends using the generic package for configuring all objects in the above mentioned packages. See section [14.2](#) for more information about generic methods. See section [12.4](#) for more information about the 5620 SAM equipment object hierarchies.

15.3 Generic NE profiles

The 5620 SAM provides limited management support of generic NEs, which are non-Alcatel-Lucent devices. These devices can be discovered and managed by the 5620 SAM, however configuration of the physical components of the node is not supported. See the *5620 SAM User Guide* for more information about generic NE support.

Before a generic NE can be discovered by the 5620 SAM, a generic NE profile has to exist. An OSS can be used to create these profiles. The package `genericne` is used for the management of generic element profiles and interfaces. The profiles define parameters used to manage non-native network elements.

A generic NE profile includes the following elements:

- the device MIB system object ID
- regular-expression strings that specify the format of prompts and commands
- the SNMP trap management configuration
- interfaces that can be specified as the endpoints of 5620 SAM physical links



Note — To manage generic NE profiles, the OSS user requires the `genericne` scope of command role.

15.4 Generic NE profile parameter configuration

The following configuration parameters outline some of the parameters from the `genericne.GenericNEProfile` class required to develop an SML request that can be used to create a generic NE profile. It does not include the parameters for configuring trap management and interface types.

- product name - used to specify the NE Type (mandatory on create)
- description (optional)
- system object ID (mandatory on create). The system object ID can be one of the following types:
 - system ID of the network element (specifies the SNMP OID for the specific product and is unique)
 - partial system ID (to facilitate the discovery of different network element types for a family of products, for example, Cisco product the general system object ID 1.3.6.1.4.1.9.1.*)
- If the devices discovered by the profile are using the 5620 SAM script management feature, you can configure the following parameters for the CLI profile that are used for read and write CLI access:
 - `commandPrompt`
 - `DisablePagingCommand`
 - `errorIndicator`
 - `resetCommand`
 - `twoStepsLogin`

- the following CLI parameters can be configured as valid CLI commands for read and write access privileges:
 - readLoginPrompt
 - readPasswordPrompt
 - writeAccessLoginCommand
 - enablingWriteAccessLoginPrompt
 - writeLoginPrompt
 - writePasswordPrompt

Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the advanced SDK configuration GNE package. This package has samples for creating generic NE profiles for various product types, such as Cisco and Juniper.



Note — You cannot modify the generic NE type after you create the generic NE profile. You must delete the generic NE profile and create a new profile with the generic NE type. You cannot delete the generic NE profile if any network elements have been discovered using the profile.

15.5 Device software upgrades

The 5620 SAM supports software upgrades of managed devices via the OSS interface. Software images must be extracted relative to the *installation_directory/server/nms/nodeSoftware* directory on the 5620 SAM server. Files in this directory are automatically deleted by the 5620 SAM after 1440 m, or one day.

Contact your Alcatel-Lucent OIPS technical-support representative for information about configuring this time period, if required.

16 – Network configuration management

- 16.1 Network configuration management overview 16-2**
- 16.2 Network interface configuration 16-2**
- 16.3 Workflow to configure network interfaces 16-2**
- 16.4 Static route configuration 16-3**
- 16.5 Routing protocol configuration 16-3**
- 16.6 Workflow to configure a routing protocol 16-6**
- 16.7 VRRP virtual router configuration 16-7**
- 16.8 Workflow to configure a virtual router 16-8**
- 16.9 MPLS, LSP, and service tunnel configuration 16-8**
- 16.10 Workflow to configure an MPLS path, LSP, and service tunnel 16-9**

16.1 Network configuration management overview

The 5620 SAM-O can be used to configure the routing and forwarding parameters on NEs. It uses the `netw` package to configure the contained network elements managed by the network manager.

The following network object configurations are required to establish the routing infrastructure and connectivity before services can be provisioned on the network:

- network interface
- static route
- routing protocol
- MPLS, LSP and service tunnel

16.2 Network interface configuration

The network interface represents a virtual router interface in the system. This virtual router interface is defined on a network port. While the physical connection of one device to another is through a port or channel linked by physical wire, it is the network interface that determines the IP connectivity.

The 5620 SAM-O XML interface uses the `rtr` package to configure router details. The `rtr.NetworkInterface` class is created and configured under the parent where the FDN is

`network:${systemAddress}:router-${*routingInstanceId}:ip-interface-${*id}`. It can also include the following parameters:

- an IP address and subnet mask associated to a physical port or channel
- associated L3 interface to a physical port or channel cabled to another device
- configured QoS policy. See chapter 17 for more information.
- configured routing protocols. See section 16.5 for more information.

See the *5620 SAM-O XML Reference* for more information about all of the parameters that can be configured on the classes from `rtr` package.

16.3 Workflow to configure network interfaces

The following workflow lists the high-level steps required to configure network interfaces to perform routing and forwarding. Each of the steps provide guidance on the type of XML request required to perform a particular configuration action.

Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the advanced SDK configuration interfaces package.

- 1 Create and configure the network interfaces that are associated with network ports. The XML configuration request creates and configures the interfaces as follows:
 - a Assign a name to the interface.
 - b Associate IP address, a network port, and QoS policy settings to the interface.
- 2 As required, create static routes. See section 16.4 for more information.
- 3 As required, configure a routing policy. See chapter 17 for more information.
- 4 Configure and enable the associated routing protocol. See section 16.5 for more information.

16.4 Static route configuration

Static routes represent the type of routing characterized by the absence of communication between routers regarding the current topology of the network. These routes represent a fixed route to a fixed destination, through a specific next hop on the nodes in the network.

As OSS application can use the 5620 SAM-O interface to create and configure static routes using the `rtr.StaticRoute` class. Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the advanced SDK configuration static routes package.

16.5 Routing protocol configuration

A routing protocol specifies how routers communicate with each other, dynamically disseminating information that enables them to select routes between nodes on a network, the choice of the route being done by routing algorithms. The 5620 SAM supports a variety of standard routing protocols.

You can use the 5620 SAM-O to enable and configure network protocol communications and connections on NEs.

Supported routing protocols and their packages include:

- BGP (bgp)
- RIP (rip)
- OSPF (ospf)
- LDP (ldp)
- IS-IS (isis)
- L2TP (l2tp)
- RSVP (rsvp)

The 5620 SAM-O also supports the following multicast protocols:

- PIM (pim)
- IGMP (igmp)
- MSDP (msdp)
- MLD (mld)

BGP

BGP is an inter-AS routing protocol. An AS is a network or a group of devices logically organized and controlled by a common network administration. BGP enables devices to exchange network reachability information. AS paths are the routes to each destination. There are two types of BGP: IBGP and EBGP.

- IBGP is used to communicate with peer devices in the same AS.
- EBGP is used to communicate with peers in different ASs.

RIP

RIP is an IGP that uses a distance-vector algorithm to determine the best route to a destination, using hop count as the deciding factor. In order for the protocol to provide complete information about routing, every device in the domain must participate in the protocol. RIP, a UDP-based protocol, updates its neighbors, and the neighbors update their neighbors. RIP directly advertises reachability information to its neighbors by sending prefix, mask, and either hop count or cost metric data.

OSPF

OSPF is a hierarchical link state protocol. OSPF is an IGP used within large ASs. OSPF routers exchange the state, cost, and other relevant interface information with neighbors after the neighbors are discovered. The information exchange enables all participating routers to establish a network topology map. Each router applies the Dijkstra algorithm to calculate the shortest path to each destination in the network. The resulting OSPF forwarding table is submitted to the routing table manager to calculate the routing table.

LDP

LDP is used to distribute labels in non-traffic-engineered MPLS applications. Routers can establish LSPs across a network by mapping network-layer routing information directly to the data link layer switched paths. After LDP distributes the labels to the LSR, the LSR assigns the label to a FEC, and then informs all other LSRs in the path about the label and how the label will switch data accordingly.

IS-IS

IS-IS is a link-state IGP that uses the shortest path first algorithm to determine a route. Routing decisions are made using the link-state information. IS-IS entities include:

- networks, which are autonomous system routing domains
- intermediate systems, which are routers, such as the 7750 SR and 7710 SR
- end systems, which are network devices that send and receive PDUs

End systems and intermediate system protocols allow devices and nodes to identify each other. The IS-IS protocol sends link state updates periodically through the network, so each device can maintain current network topology information.

L2TP

L2TP is a session-layer protocol that extends the PPP model by allowing L2 and PPP endpoints to reside on different devices that are interconnected by a PSN. L2TP extends the PPP sessions between the CPE and PPP/L2TP termination point (LNS), via an intermediate LAC.

RSVP

RSVP is a network-control protocol in the IP suite that is used for communicating application QoS requirements to intermediate transit nodes in a network. RSVP uses a soft-state mechanism to maintain path and reservation states on each node in the reservation path.

PIM

PIM is a multicast routing architecture that allows the addition of IP multicast routing on existing IP networks. PIM is unicast routing protocol independent and can be operated in the following ways:

- sparse mode
- dense mode

IGMP

The 5620 SAM-O uses the igmp package to configure the multicast IGMP routing type on IPv4 hosts and routers. IPv4 hosts and routers use IGMP to report their group memberships to neighboring multicast routers.

MSDP

MSDP is a protocol that enables multiple PIM-SM domains to communicate with each other using their own RPs. MSDP also enables multiple RPs in a single PIM-SM domain to establish MSDP mesh-groups and to synchronize information between anycast RPs about the active sources being served by each anycast RP peer. The 7750 SR and 7710 SR support MSDP.

MLD

MLD is an asymmetric protocol used by IPv6 routers to discover the presence of nodes that wish to receive multicast packets on their directly-attached links, and to discover which multicast addresses are of interest to those neighboring nodes.



Note — See the *5620 SAM User Guide* for more information about routing protocols.

16.6 Workflow to configure a routing protocol

The following workflow lists the high-level steps required to configure a BGP, RIP, OSPF, LDP, or ISIS routing instance, or a PIM or IGMP multicast instance. Each routing protocol has different configuration steps that need to be followed to get routing operational in the network.

Each step in this workflow represents an XML configuration request. Contact your Alcatel-Lucent OIPS technical support representative for samples of these XML scripts from the advanced SDK configuration protocols and router packages.

- 1 For a BGP routing instance:
 - i Configure an AS number.
 - ii Configure a number for the confederation-autonomous system if your configuration requires confederations.
 - iii Enable BGP on a router.
 - iv Configure global-level BGP elements.
 - v Configure a BGP confederation.
 - vi Configure a peer group-level BGP.
 - vii Configure a peer-level BGP.
- 2 For a RIP routing instance:
 - i Enable RIP on a router.
 - ii Configure a global-level RIP.
 - iii Configure a group-level RIP.
- 3 For an OSPF routing instance:
 - i Enable OSPF on a router.
 - ii Configure OSPF elements.
 - iii Configure an OSPF area.
 - iv Add an L3 interface to the OSPF area.

- 4 For an LDP routing instance:
 - i Enable LDP on a router.
 - ii Configure global-level LDP elements.
 - iii Configure LDP interfaces.
 - iv Configure LDP targeted peers.
- 5 For an ISIS routing instance:
 - i Enable ISIS on a router.
 - ii Configure router-wide ISIS elements.
 - iii Configure ISIS NET addresses.
 - iv Configure ISIS interfaces.
- 6 For PIM routing instance:
 - i Enable PIM.
 - ii Configure global-level PIM.
 - iii Configure a PIM source-specific multicast group.
 - iv Configure a PIM static rendezvous point.
 - v Configure a PIM interface.
 - vi Configure the PIM candidate rendezvous point.
 - vii Configure the PIM anycast rendezvous point.
- 7 For IGMP routing instance:
 - i Enable IGMP.
 - ii Configure IGMP.
 - iii Configure an IGMP interface.

16.7 VRRP virtual router configuration

VRRP allows the creation of a redundant routing system that takes over packet transmissions on a common LAN segment when a router fails. VRRP designates alternative routing paths in the form of a virtual router, without changing the IP address or MAC address of a protected router. The 5620 SAM-O supports the creation of virtual routers on IES services and core network LAN services.

With VRRP, a protected router owns the IP address of the virtual router. In the role of master, the owner router normally forwards packets to hosts on the default gateway. If the owner router fails, the virtual router, which has the same IP address as the owner, shifts packet-forwarding responsibilities to a designated backup router. The backup router then becomes the master router and forwards packets on the virtual router IP address.

16.8 Workflow to configure a virtual router

The following workflow lists the high-level steps required to configure a virtual router on a core network LAN or an IES. Each step represents an XML configuration request. Contact your Alcatel-Lucent OIPS technical support representative for samples of these requests from the advanced SDK configuration protocols packages.

This workflow assumes the following:

- an L3 interface primary IP address is available for the network or IES service.
 - each IP interface must have a primary IP address.
- 1 Create a global VRRP priority-control policy for non-owner VRRP instances.
 - 2 Distribute the global VRRP priority-control policy to a non-owner VRRP router.
 - 3 Create a virtual router for a core network LAN, or an IES service.
 - 4 As required, create an owner VRRP instance in the virtual router.
 - 5 As required, create a non-owner VRRP instance in the virtual router.

16.9 MPLS, LSP, and service tunnel configuration

After your routing protocols have been configured, the next steps are to establish the network's data forwarding transport infrastructure that will eventually be used to carry services across the entire network. MPLS paths, LSPs, and service tunnels need to be created and configured to achieve this.

The 5620 SAM-O XML interface uses the mpls package for the configuration and provisioning of MPLS paths and LSPs on managed routers and it uses the svt package to configure service tunnels. See the *5620 SAM-O XML Reference* for more information about these packages.

MPLS

MPLS is a data-carrying mechanism that uses one or more routing protocols to forward packets. MPLS can be used as the underlying transport mechanism for service tunnels. For MPLS to be used as such, an MPLS mesh and an LSP mesh must be created before the tunnel is created.

LSP

An LSP is a path through an MPLS network that is set up based on criteria in a forwarding equivalency class, or FEC. LSPs are unidirectional; they enable the label switching of a packet through an MPLS network from one endpoint to another. Bidirectional communication through an MPLS network requires the configuration of an LSP in the opposite direction.

Service tunnels

A service tunnel is an entity used to uni-directionally direct traffic from one device to another device. The service tunnel is provisioned to use a specific encapsulation method, such as GRE or MPLS, and the services are then mapped to the service tunnel. The most common type of tunnel used in the 5620 SAM is a service distribution point binding. Service tunnels originate on an SDP on a source NE and terminate at a destination NE.



Note — See the *5620 SAM User Guide* for more information about MPLS, LSPs, and service tunnels.

16.10 Workflow to configure an MPLS path, LSP, and service tunnel

The following workflow lists the high-level steps required to configure an MPLS path, LSP, and service tunnel. Each step represents an XML configuration request. Contact your Alcatel-Lucent OIPS technical support representative for samples of these requests from the advanced SDK configuration servicetunnels packages.

This workflow assumes that at least one IGP is enabled and configured on all participating NEs and includes the system interface. See section 16.5 for more information.

- 1 Enable MPLS routing on all applicable routers and L3 interfaces.
- 2 Assign a network interface to an MPLS instance.
- 3 Create a full mesh of MPLS path which is the route to be used by an LSP, between the routers using the L3 interfaces.
- 4 Create a full mesh of LSPs and bind to the MPLS paths.
- 5 Create service tunnels between all routers that have access ports. The service tunnels direct traffic from one router to another. You must create service tunnels in both directions because service tunnels are unidirectional.
 - a As required, create a GRE service tunnel.
 - b As required, create an MPLS LSP service tunnel.

17 – Policy configuration management

- 17.1 Policy configuration overview 17-2**
- 17.2 Workflow to configure ACL IP and MAC filter policies 17-7**
- 17.3 Workflow to configure routing management policies 17-7**

17.1 Policy configuration overview

The 5620 SAM-O supports the template-based creation of policies. There are three types of policies:

- service management
- routing management
- network management

The 5620 SAM-O supports the creation and modification of policies using the following packages:

- | | |
|--------------|------------------|
| • accounting | • policy |
| • aclfilter | • portscheduler |
| • acl | • qos |
| • aengr | • rp |
| • aingr | • snmp |
| • atmpolicy | • slope |
| • file | • squeue |
| • mediation | • svq |
| • multicast | • tod |
| • niegr | • vs |
| • nqueue | • telcoaclfilter |
| • pae802_1x | • telcoqos |



Note — Alcatel-Lucent recommends using the generic package for configuring all objects. See section 14.2 for more information about generic methods. See section 12.4 for more information about the logical object hierarchy for all policy objects.

The abstract policy class and the policy manager control all QoS policies. All other QoS policies inherit from classes in this package.

The qos policy class defines the basic types related to quality of service, such as enumerations, that are common to and used by other QoS-related packages, such as acl, aclfilter, file, niegr, and vs. There are no methods associated with this package.

Service management policies

Service management policies specify how service traffic is handled by network resources such as interfaces, ports, daughter cards, and circuits. Service management policies are globally and seamlessly distributed to routers when they are used by resources on the router. You can also manually distribute the policies to the routers. Service management policies include:

- | | |
|------------------|-----------------|
| • access ingress | • network queue |
| • access egress | • scheduler |
| • network policy | • ACL IP filter |
| • slope | • MAC IP filter |

See [“Workflow to configure ACL IP and MAC filter policies”](#) for more information.

aingr package

Access ingress policies are applied to access interfaces and specify QoS on ingress.

Access ingress policies define ingress service forwarding class queues and map flows to those queues. You cannot delete the following queues when you create an access ingress policy:

- default unicast traffic
- default multipoint traffic

The queues exist within the definition of the policy. The queues are only instantiated in hardware when the policy is applied to an access interface. In the case where the service does not have multipoint traffic, the multipoint queue is not instantiated. In the simplest access ingress policy, all traffic is treated as a single flow and mapped to a single queue, and all flooded traffic is treated with a single multipoint queue.

The required access ingress policy elements include:

- a unique access ingress policy ID
- at least one default unicast forwarding class queue
- at least one multipoint forwarding class queue

The optional access ingress policy elements include:

- additional unicast queues up to a total of eight for each of the eight forwarding classes
- additional multipoint queues up to three for each forwarding class for each type of multipoint traffic (broadcast, multicast, and destination unknown unicast)
- QoS policy match criteria to map packets to a forwarding class

Each queue can have unique queue parameters to allow individual policing and rate shaping of the flow mapped to the forwarding class. Mapping flows to forwarding classes is controlled by comparing each packet to the match criteria in the policy. There is one default access ingress policy. The default policy gives all traffic equal priority with the same chance of being sent or dropped during periods of congestion.

You can use forwarding subclasses for additional access ingress packet classification. You can associated one or more subclasses with each forwarding class. The designations for forwarding subclasses are the same as the designation used for the forwarding classes. Each subclass assumes the behavior of the parent forwarding class. The two-tier class designation provides enhanced classification for access ingress QoS policies.

aengr package

Access egress policies are applied to access egress interfaces and specify QoS on egress.

Access egress policies define egress service queues and map forwarding class flows to queues. In the simplest access egress policy, all forwarding classes are treated as a single flow and mapped to a single queue.

The required access egress policy elements include:

- a unique access egress policy ID
- at least one defined default queue

The optional egress policy elements include:

- additional queues up to eight separate queues for each of the eight supported forwarding classes
- IEEE 802.1p priority value remarking based on forwarding class

Each queue in a policy is associated with one or more of the supported forwarding classes. Each queue can have its own queue parameters that allow individual rate shaping of the forwarding classes mapped to the queue. More complex service queuing models are supported on the managed devices where each forwarding class is associated with a dedicated queue.

niegr package

Network policies are applied to network interfaces and specify QoS on egress and ingress.

On ingress, a network policy maps incoming DSCP and EXP values to forwarding class and profile state for traffic received from the core network. On egress, the policy maps forwarding class and profile state to DSCP and EXP values for traffic to be transmitted into the core network.

Routing management policies

Routing management policies control the size and content of the routing tables, advertised routes, and the recommended route to each destination. You can use routing management policies to configure:

- AS paths
- communities
- dampings
- policy statements
- prefix lists

See [“Workflow to configure routing management policies”](#) for more information.

Network management policies

Network management policies specify how the 5620 SAM communicates with network resources, handles alarms, manages billing statistics, and stores information. Network management policies include:

- alarm
- file
- accounting

- mediation
- poller

Policy distribution mode

Global policies are policies that you create using the 5620 SAM-O or the 5620 SAM GUI, or that are created through node discovery or CLI. Local policies are instances of global policies that are assigned to individual NEs. Depending on the distribution mode configuration of a local policy, when you modify a global policy, all local instances of the policy can be automatically updated. This ensures that all instances of the policy in the network are synchronized.

Table 17-1 describes the policy distribution modes.

Table 17-1 Distribution mode

Option	Option description
Sync With Global	The local policy is synchronized with the global policy at all times. The local instance cannot be modified.
Local Edit Only	You can modify the local instance only, which affects the associated network element. Changes to the global policy do not affect the local policy unless a synchronization operation is manually performed.

When you distribute a global policy, local policies using the Sync With Global distribution mode allow the NE to receive the policy.



Note — Local policies using the Local Edit Only distribution mode do not allow the NE to receive the distribution of a global policy. You must ensure that the policy distribution mode for the local policy is set to Sync With Global if you want the NE to receive the distribution of a global policy.

The `isPolicyDiscoveredInLocalEditOnlyMode` flag in the `nms-server.xml` file determines the distribution mode to which a policy is set when it is created through node discovery or CLI. By default, the flag is set to false and local policies that are created through NE discovery or CLI are set to Sync With Global distribution mode.

```
<policyConfig isPolicyDiscoveredInLocalEditOnlyMode="false"/>
```

If you set the `isPolicyDiscoveredInLocalEditOnlyMode` flag to true in the `nms-server.xml` file, local policies that are discovered from the network or created through CLI are set to Local Edit Only mode.

Use the `setDistributionModeToLocalEditOnly` method to change a specific policy to Local Edit Only distribution mode. Use the `setDistributionModeToSyncWithGlobal` method to change a specific policy to Sync With Global distribution mode and automatically synchronize with the previous released global policy. See the *5620 SAM-O XML Reference* for more information.



Note — You must have the appropriate permissions to use these methods. See the *5620 SAM User Guide* for more information about 5620 SAM user groups and the associated scope of command roles and permissions.

Policy configuration mode

The `isPolicyConfigurationModeUsed` flag in the `nms-server.xml` file determines the configuration mode in which a policy is created through the 5620 SAM-O. By default, the flag is set to false, or released mode.

```
<policyOssiConfig isPolicyConfigurationModeUsed="false" />
```

Global policies that are created by local policy discovery are in draft mode.

When the `isPolicyConfigurationModeUsed` flag is set to false, global policies that are configured through the 5620 SAM-O are automatically released and distributed to local policies. If you update a policy, regardless of the configuration mode, the 5620 SAM-O accepts the changes and redistributes the policy.

When you distribute a global policy, local policies using the Sync With Global distribution mode allow the NE to receive the policy.



Note — Local policies using the Local Edit Only distribution mode do not allow the NE to receive the distribution of a global policy. You must ensure that the policy distribution mode for the NE is set to Sync With Global if you want the NE to receive the distribution of a global policy.

When the `isPolicyConfigurationModeUsed` flag is set to false and you attempt to distribute a policy that is in draft mode—created or modified from the 5620 SAM GUI and not yet released—the policy is distributed to the network and the local instances are updated.

If you set the `isPolicyConfigurationModeUsed` flag to true in the `nms-server.xml` file, all global policies are created in draft mode. If the policy is later released, the 5620 SAM-O returns the policy to draft mode when any subsequent modifications to the policy are made. When a draft policy is bound to a service and a local policy does not exist, the 5620 SAM deploys the backup policy, or previously released policy, and distributes the policy to the associated site. If a backup policy does not exist, an exception is raised.

Use the `setConfigurationModeToReleased` method to release and distribute a draft policy. See the *5620 SAM-O XML Reference* for more information.



Note — You must have the appropriate permissions to use this method. See the *5620 SAM User Guide* for more information about 5620 SAM user groups and the associated scope of command roles and permissions.

17.2 Workflow to configure ACL IP and MAC filter policies

Use the following steps to configure ACL IP and MAC filter policies.

- 1 Create the filter. Contact your Alcatel-Lucent OIPS technical support representative for information about how to obtain a sample request to find ACL IP filters.
- 2 Distribute the filter. Contact your Alcatel-Lucent OIPS technical support representative for information about how to obtain a sample request to distribute an ACL IP filter.
- 3 Synchronize the filter. Contact your Alcatel-Lucent OIPS technical support representative for information about how to obtain a sample request to synchronize an ACL IP filter.

17.3 Workflow to configure routing management policies

Use the `policy.Manager.configure` method, as defined in `policyMethods.xsd` and `policyTypes.xsd` files, and the following steps to configure routing management policies.

- 1 Configure the following parameters for AS paths:
 - i Specify a path name. The range is 1 to 32 characters.
 - ii Specify a description, if required. The range is 0 to 80 characters.
 - iii Specify a regular expression. The range is 1 to 80 characters.
- 2 Configure the following parameters for communities:
 - i Specify a community name.
 - ii Configure the community members for the community defined in step [2 i](#). You can configure multiple community members.
- 3 Configure the following parameters for Dampings:
 - i Specify a damping name. The range is 1 to 32 characters.
 - ii Specify a half-life value. The range is 0 to 45.
 - iii Specify a reuse value. The range is 0 to 20 000.
 - iv Specify a suppress value. The range is 0 to 20 000.
 - v Specify a maximum suppression value. The range is 0 to 720.

- 4 Configure the following parameters for prefix lists:
 - i Specify a prefix list name.
 - ii Specify the prefix list members to be added to the prefix list defined in step 4 i. You can configure multiple prefix list members. Configure the following parameters for each member:
 - Specify a prefix. Enter an IP address.
 - Specify a mask. Enter a network mask in bitmask format. The range is 0 to 32.
 - Specify a type. The options are: Exact, Longer, Through, or Range.
 - Configure the Begin Length and Through Length parameters. The availability of these parameters depends on the option selected for the Type parameter. The range for both parameters is 24 to 32. The value is in bitmask format.
- 5 Configure the following parameters for policy statements:
 - i Specify a policy statement name. The range is 1 to 32 characters.
 - ii Specify a description. The range is 0 to 80 characters.
 - iii Specify a default action. The options are: Reject or Accept.

- iv Specify policy entries to add to the policy statement. You can configure multiple policy entries. Configure the following parameters for each policy entry:
 - Specify an entry ID. The range is 131 072.
 - Specify a description. The range is 0 to 80 characters.
 - Specify an action. The options are: Reject or Accept.
 - Specify the following parameters for the From Criteria:
 - AS Path Name. The range is 0 to 32 characters.
 - Protocol. The options are: None, Direct, Static, BGP, ISIS, OSPF, RIP, Aggregate, BGP-VPN, IGMP, or PIM.
 - Community List Name. The range is 0 to 32 characters.
 - Interface Name. The range is 0 to 32 characters.
 - ISIS Route Level. The options are: 0, 1, or 2.
 - ISIS External Route. The options are: True or False.
 - OSPF Route Type. The options are: 0, 1, or 2.
 - OSPF Area. Enter an IP address.
 - OSPF Origin. The options are: None, IGP, EGP, or Incomplete.
 - OSPF Area Set. The options are: True or False.
 - Neighbor IP Address. Enter an IP address.
 - Neighbor Prefix List Name. The range is 0 to 32 characters.
 - Prefix Lists. You can enter up to five prefix lists. The range is 0 to 32 characters.
- v Specify the following parameters for the To Criteria:
 - Protocol. The options are: None, BGP, ISIS, OSPF, RIP, or BGP-VPN.
 - ISIS Route Level. The options are: 0, 1, or 2.
 - Neighbor IP Address. Enter an IP address.
 - Neighbor Prefix List Name. The range is 0 to 32 characters.

6 Construct a valid SOAP request.

7 Send the request.

Contact your Alcatel-Lucent OIPS technical support representative for information about how to obtain a sample request to configure a policy.

18 – Service configuration management

- 18.1 Service configuration overview 18-2
- 18.2 Workflow to configure a mirror service 18-16
- 18.3 Workflow to configure an optical transport service 18-17
- 18.4 Composite services 18-17
- 18.5 Workflow to configure a composite service 18-22
- 18.6 IGMP snooping 18-22
- 18.7 Workflow to configure IGMP snooping 18-22
- 18.8 DHCP relay configuration 18-23
- 18.9 Workflow to configure DHCP relay 18-23
- 18.10 IPsec management 18-23
- 18.11 Workflow to configure IPsec – option A 18-25
- 18.12 Workflow to configure IPsec – option B 18-25
- 18.13 Customer and residential subscriber configuration overview 18-26
- 18.14 Workflow to configure and manage residential subscribers 18-27

18.1 Service configuration overview

After the network devices, protocols, IP/MPLS components, and policies have been configured and established, the next step is for network providers to provision services to be used by end users.

The service is a means of transport for the application content and is owned by a service customer typically called the service provider. See section 18.13 for more information about how to use 5620 SAM-O to create and configure the customers that require services for their end users and residential subscribers.

OSS applications can use the 5620 SAM-O to create, configure, and manage services based on the XML schema. The user privileges associated with the OSS user define the ability to configure 5620 SAM objects.

The 5620 SAM-O XML interface uses the service.serviceManager (svc-mgr) package to manage services. Configuration operations include creating, deleting, and configuring services on sites and routers.

The 5620 SAM supports the following service types:

- VLL
- VPLS/HVPLS/MVPLS
- IES
- VPRN
- VLAN
- Mirror
- Optical transport service
- Object life cycle

VLL

The 5620 SAM-O uses the vll package with the following packages to configure a VLL service based on the following service types:

- Apipe, or ATM VLL service (apipe)
- Epipe, or Ethernet VLL service (epipe)
- Fpipe, or frame relay VLL service (fpipe)
- Ipipe, or IP internetworking VLL service (ipipe)
- Cpipe, or circuit emulation VLL service (cpipe)



Note — For the 9500 MPR, Epipe, Cpipe and Apipe services use the mpr package for creating VLL services.

The 5620 SAM supports the provisioning of L2 VLL services on edge devices. A VLL is a pipe that connects access interfaces. A VLL that connects access interfaces on one router (or site) is called a local VLL service. A VLL that connects access interfaces on two routers (or sites) is called a distributed VLL service.

For the distributed VLL service, subscriber data enters the service through two access interfaces on different edge devices. The VLL is transported across an IP and/or IP/MPLS provider core network in circuits that are carried by service tunnels. Service tunnels are created using GRE or MPLS LSPs. You can configure and specify circuits for the VLL service during service creation. You must create service tunnels before you create the VLL service.

Packets that arrive at an edge device are associated with a VLL service based on the access interface on which they arrive. An access interface is uniquely identified by the:

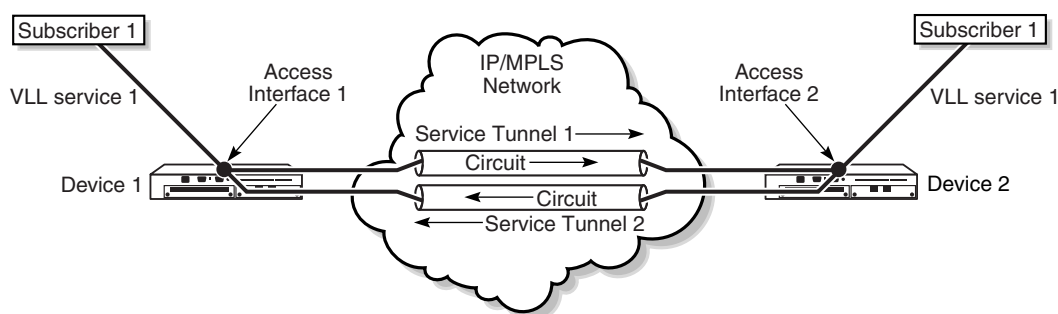
- physical port or POS port and channel
- encapsulation type
- encapsulation identifier (if required, depending on encapsulation type)

See the *5620 SAM User Guide* for more information about VPLS management.

Sample VLL creation and configuration request development

Figure 18-1 shows a sample VLL service.

Figure 18-1 Sample VLL service



17237

Assuming the core IP/MPLS network and service tunnels have already been configured, the following high-level configuration steps are required to develop an XML request that creates and configures a basic VLL service similar to VLL service 1 in Figure 18-1 with sites, L2 access interfaces and policies:

- Configure policies, as required. Policies should be created and defined prior to creating the service. Typical policies that are applied to be part of VLL services include QoS policies like Access Ingress, Access Egress, Scheduler Policies, Filter Policies, Accounting Policies, and ANCP Policies. See chapter 17 for more information.
- Configure ports as access ports and encapsulation types, as required. See chapter 16 for more information.
- Configure service tunnels, as required. (Service tunnel 1 and 2). See chapter 16 for more information.
- Create and configure customer/subscriber. (Subscriber1). See section 18.13 for more information.

- Create a VLL service that will include two sites that contain L2 access interfaces for each site. The XML request will include the following:
 - Customer
 - Two sites (Device 1 and 2) for the VLL service
 - Access interfaces (1 and 2) for each site and configure the following—port or channels and encapsulation values (as required), and the policies (as required)

Below are the minimum XML parameter tags required to develop an XML request to create the basic VLL service described above.

The `genericObject.configureInstance` method is used to create this service. The request response uses the `<objectFullName>` element to identify the new service.

Service creation parameters

- `distinguishedName` - svc-mgr
- `childConfigInfo` - creates and configures the service parameters below based on the specific VLL type.
 - `<subscriberPointer>` - assign Subscriber1
 - `<serviceId>` - assign unique serviceId
 - `<transportPreference>` - specify transport GRE/MPLS or any (optional)

Site creation parameters

- `children-Set` - creates and configures the sites from the service based on the specific VLL type.
 - `<siteId>`
 - `<administrativeState>` - used to turn the service up

L2 access interface parameters

- `children-Set` - creates and configures the `<vll.L2AccessInterfaces>` (SAPS) from the sites.
 - `<administrativeState>`
 - `<portPointer>` - assign access port
 - `<innerEncapValue>/<outerEncapValue>` - assign encapsulation values
 - `<ingressPolicyObjectPointer>` - assigns the ingress Policy

See the *5620 SAM-O XML Reference* and *5620 SAM-O XML Schema* for more information about classes, properties, and methods to create VLL services.

Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the Advanced SDK - Configuration - Services package. This package has samples for creating VLL services for all of the VLL service types.

VPLS/HVPLS/MVPLS

The 5620 SAM-O uses `vpls` and `mvpls` packages to configure VPLS/HVPLS services and MVPLS services respectively. Two variants of VPLS are HVPLS (Hierarchical VPLS) and MVPLS (Management VPLS).

A VPLS is a class of virtual private network multipoint L2 service that allows multiple customer sites to be connected in a single bridged domain contained within the service provider-managed IP/MPLS network. Customer sites in the VPLS appear to be on the same LAN, even if the sites are geographically dispersed.

A VPLS that spans a single site is called a local VPLS. In a local VPLS, subscriber data enters the service through multiple access interfaces on a single PE device. Circuit provisioning is not required for the local VPLS.

A VPLS can span single or multiple sites. A VPLS that spans a single site is called a local VPLS. In a local VPLS, subscriber data enters the service through multiple access interfaces on a single PE device. A VPLS that spans multiple sites is called a distributed VPLS. In a distributed VPLS, customer data enters the service using two or more interfaces on different PE devices. The VPLS is transported by service circuits over an IP/MPLS provider core network that is carried by service tunnels.

An HVPLS can be created by enhancing the VPLS core mesh with a spoke SDP binding that is connected to another site in the same VPLS. It eliminates the need for a full mesh of virtual circuits between devices in the VPLS.

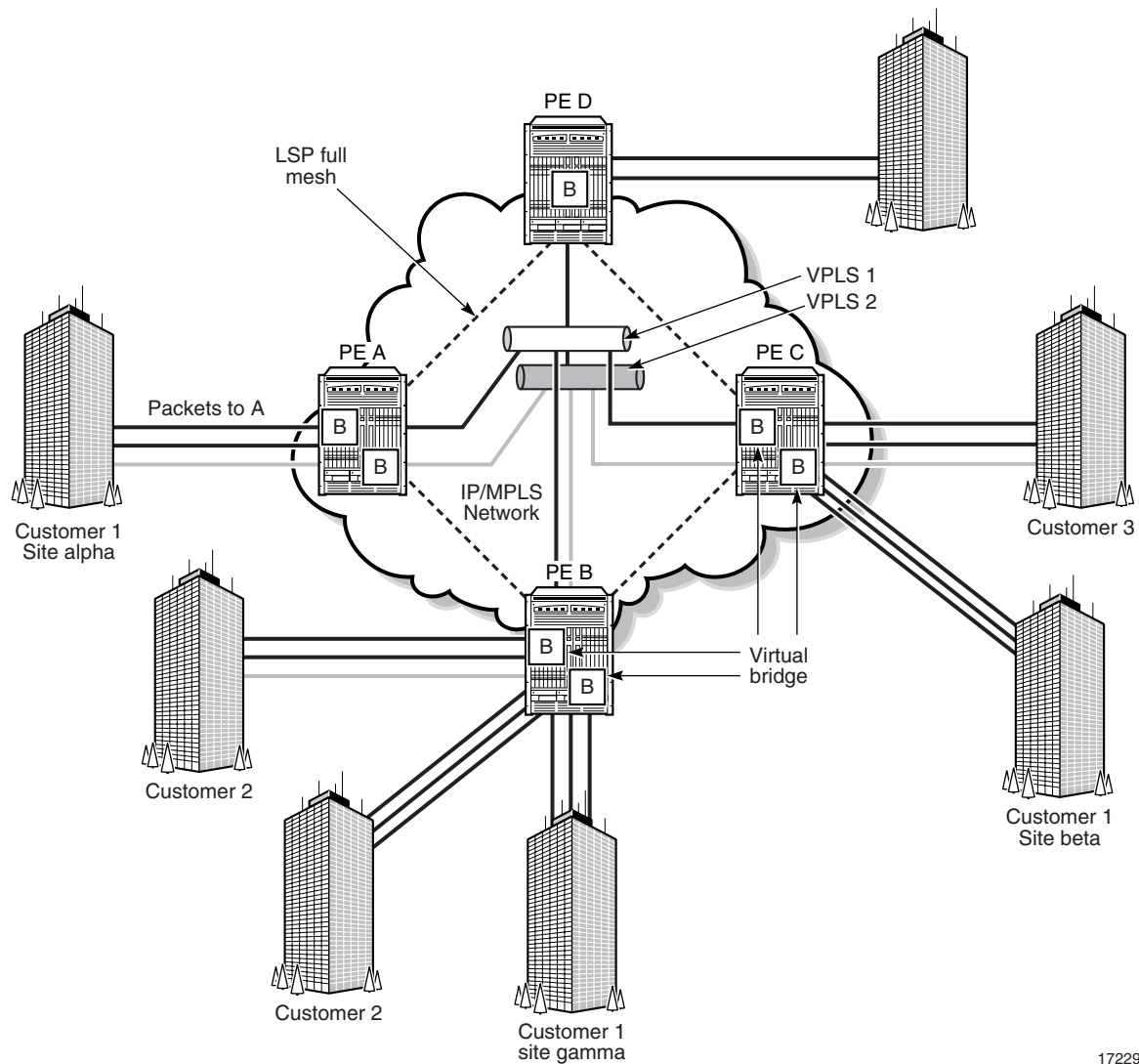
An MVPLS is required to remove topology loops when redundant spoke SDPs or L2 access interfaces have been created for HVPLS or VPLS configurations.

See the *5620 SAM User Guide* for more information about VPLS management.

Sample VPLS creation and configuration request development

Figure 18-2 shows a sample VPLS configuration.

Figure 18-2 Sample VPLS



17229

Assuming the core IP/MPLS network and service tunnels have already been configured, the following high-level configuration steps are required to develop an XML request that creates and configures a basic VPLS service similar to the VPLS service in Figure 18-2 with many sites, L2 access interfaces and policies:

- Configure policies, as required. Policies should be created and defined prior to creating the service. Typical policies that are applied to be part of VPLS services include QoS policies like Access Ingress, Access Egress, Scheduler Policies, Filter Policies, Accounting Policies, DoS protection Policies, and ANCP Policies. See chapter 17 for more information.
- Configure ports as access ports and encapsulation types, as required. See chapter 16 for more information.
- Configure service tunnels or SDPs, as required, in both directions for each tunnel. If the VPLS is a distributed VPLS, configure mesh SDP bindings to connect all sites. See chapter 16 for more information.

- Create and configure customer/subscriber. See section 18.13 for more information.
- Create a VPLS service that can include many sites that contain L2 access interfaces for each site. The XML request could include the following:
 - Customer
 - The sites for the VPLS service
 - Access interfaces for each site and configure the following and repeat for each site—port or channels and encapsulation values (as required), and the policies (as required)

Below are the minimum XML parameter tags requires to develop an XML request to create the basic VPLS service described above.

The `genericObject.configureInstance` method is used to create this service. The request response uses the `<objectFullName>` element to identify the new service.

Service creation parameters

- `distinguishedName` - svc-mgr
- `childConfigInfo` - creates and configures the service parameters below based on the `<vpls.Vpls>` class.
 - `<subscriberPointer>` - assign Subscriber1
 - `<serviceId>` - assign unique serviceId
 - `<transportPreference>` - specify transport GRE/MPLS or any (optional)

Site creation parameters

- `children-Set` - creates and configures the `<vpls.Site>` class.
 - `<siteId>`
 - `<administrativeState>` - used to turn the service up

L2 access interface parameters

- `children-Set` - creates and configures the `<vpls.L2AccessInterfaces>` (SAPS) from the sites.
 - `<administrativeState>`
 - `<portPointer>` - assign access port
 - `<innerEncapValue>/<outerEncapValue>` - assign encapsulation values
 - `<ingressPolicyObjectPointer>` - assigns the ingress Policy

See the *5620 SAM-O XML Reference* and *5620 SAM-O XML Schema* for more information about classes, properties, and methods to create VPLS services.

Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the Advanced SDK - Configuration - Services package. This package has samples for creating VPLS services.

IES

The 5620 SAM-O interface uses the `ies` package to create and manage IES services.

An IES is a routed connectivity service where the subscriber communicates with an IP router interface to send and receive Internet traffic. The IP router interface is an L3 interface.

IES allows customer-facing IP interfaces in the same routing instance to be used for connectivity in service network core routing. IES requires that the IP address scheme that is used by the subscriber is unique among other provider address schemes and potentially the entire Internet.

When you configure or modify a service, you can:

- add access interfaces
- configure or modify existing access interfaces

Packets that arrive at the edge 7750 SR are associated with an IES based on the access interface on which they arrived. An access interface is uniquely identified by the:

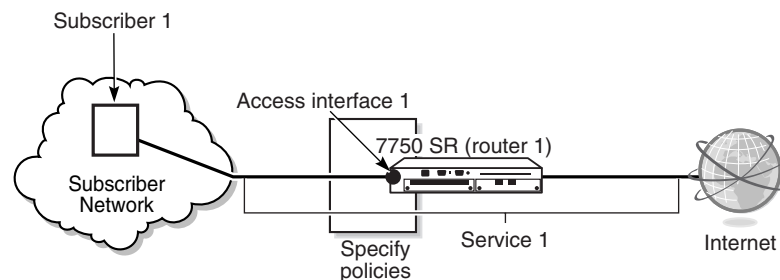
- port
- service ID
- IP address

See the *5620 SAM User Guide* for more information about IES management.

Sample IES creation and configuration request development

Figure 18-3 shows a sample IES.

Figure 18-3 Sample IES



17233

The following high-level tasks are required to configure a basic IES service similar to Service 1 in Figure 18-3. You do not need an IP/MPLS network for IES services.

- Configure policies, as required. Policies should be created and defined prior to creating the service. Typical policies that are applied to be part of VPLS services include QoS policies like Access Ingress, Access Egress, Scheduler Policies, Filter Policies, Accounting Policies, and ANCP Policies. See chapter 17 for more information.
- Create and configure customer/subscriber. See section 18.13 for more information.

- Create an IES service that includes one site that contains a source L3 access interfaces. The XML request could include the following:
 - Customer - Subscriber 1
 - The site - router 1
 - Create and configure the access interface for the site and configure the following—port or channels and encapsulation values (as required), the policies (as required), and the router IP address.

Below are the minimum XML parameter tags requires to develop an XML request to create the basic IES service described above.

The `genericObject.configureInstance` method is used to create this service. The request response uses the `<objectFullName>` element to identify the new service.

Service creation parameters

- `distinguishedName` - svc-mgr
- `childConfigInfo` - creates and configures the service parameters below based on the `<ies.Ies>` class.
 - `<subscriberPointer>` - assign Subscriber
 - `<serviceId>` - assign unique serviceId

Site creation parameters

- `children-Set` - creates and configures the `<ies.Site>` class.
 - `<siteId>`
 - `<administrativeState>` - used to turn the service up
 - `<routingInstanceId>` - default routing instance on the router

L3 access interface parameters

- `children-Set` - creates and configures the `<ies.L3AccessInterfaces>` (SAP) from the site.
 - `<administrativeState>` - turn up SAP
 - `<interfaceId>` - Interface id
 - `<portPointer>` - assign access port
 - `<innerEncapValue>/<outerEncapValue>` - assign encapsulation values
 - `<ingressPolicyObjectPointer>` - assigns the ingress Policy
 - `children-Set` - creates and configures `<rtr.VirtualRouterIpAddress>`
 - `<ipAddress>` - IP address of the router

See the *5620 SAM-O XML Reference* and *5620 SAM-O XML Schema* for more information about classes, properties, and methods to create IES services.

Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the Advanced SDK - Configuration - Services package. This package has samples for creating IES services.

VPRN

The 5620 SAM-O interface uses the `vprn` package to create and manage VPRN services.

The 5620 SAM supports the creation of VPRN services using managed devices as a PE and P router. VPRNs are also called IP VPNs or BGP/MPLS VPNs. RFC 2547bis defines VPRNs and describes a method for forwarding data and distributing routing information across an IP/MPLS provider core network.

A VPRN service consists of CE routers or devices that are connected to PE routers. PE routers connected to P routers transport data across the IP/MPLS provider core network in service tunnels.

Packets that arrive at an edge 7750 SR are associated with a VPRN service based on the access interface on which they arrive. An access interface is uniquely identified by the:

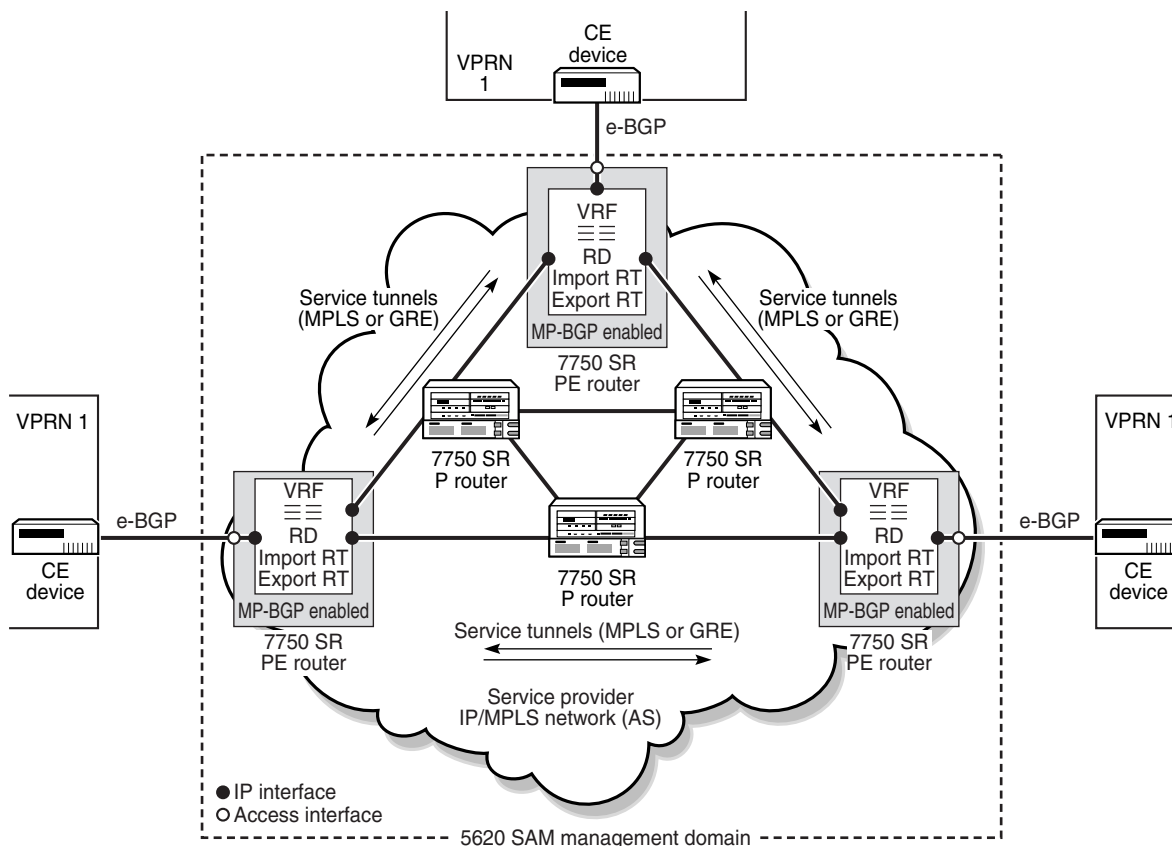
- physical Ethernet port or POS port and channel
- encapsulation type
- encapsulation identifier, if required

See the *5620 SAM User Guide* for more information about VPRN management.

Sample VPRN creation and configuration request development

Figure 18-4 shows a sample VPRN service configuration.

Figure 18-4 Sample VPRN



17333

Assuming the core IP/MPLS network and service tunnels have already been configured, the following high-level tasks are required to develop an XML request that creates and configures a basic VPRN service similar to VPRN 1 in Figure 18-4 with many sites (P and PE routers), L3 access interfaces, VRFs, IP interfaces and policies.

- Configure polices, as required. Policies should be created and defined prior to creating the service. Typical policies that are applied to be part of VLL services include Routing Policies, QoS policies like Access Ingress, Access Egress, Scheduler Policies, Filter Policies, Accounting Policies, and ANCP Policies. See chapter 17 for more information.
- Configure ports as access ports and encapsulation types, as required. See chapter 16 for more information.
- Configure service tunnels. See chapter 16 for more information.
- Create and configure customer/subscriber. See section 18.13 for more information.
- Create a VPRN service that includes many sites that contain L3 access interfaces for each site. The XML request could include the following:
 - Customer
 - Specify the site and configure the following—routing properties, VRF targets, import and export routing policies, include and specify L3 access interfaces (SAP) for each site and configure the following and repeat for each SAP: port and encapsulation values (as required), the policies (as required), and the local IP address.

Below are the minimum XML parameter tags requires to develop an XML request to create the basic VPRN service described above.

The `genericObject.configureInstance` method is used to create this service. The request response uses the `<objectFullName>` element to identify the new service.

Service creation parameters

- `distinguishedName` - svc-mgr
- `childConfigInfo` - creates and configures the service parameters below based on the `<vpn.Vpn>` class.
 - `<subscriberPointer>` - assign Subscriber
 - `<serviceId>` - assign unique serviceId

Site creation parameters

- `children-Set` - creates and configures the `<vpn.Site>` class.
 - `<siteId>`
 - `<administrativeState>` - used to turn the service up
 - `<routingInstanceId>` - default routing instance on the router

L3 access interface parameters

- `children-Set` - creates and configures the `<vpn.L3AccessInterfaces>` (SAPs) from the vpn sites.
 - `<administrativeState>` - turn up SAP
 - `<interfaceId>` - Interface id
 - `<portPointer>` - assign access port
 - `<innerEncapValue>/<outerEncapValue>` - assign encapsulation values
 - `<ingressPolicyObjectPointer>` - assigns the ingress Policy
 - `children-Set` - creates and configures `<rtr.VirtualRouterIpAddress>`
 - `<ipAddress>` - IP address of the router

See the *5620 SAM-O XML Reference* and *5620 SAM-O XML Schema* for more information about classes, properties, and methods to create VPRN services.

Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the Advanced SDK - Configuration - Services package. This package has samples for creating VPRN services.

VLAN

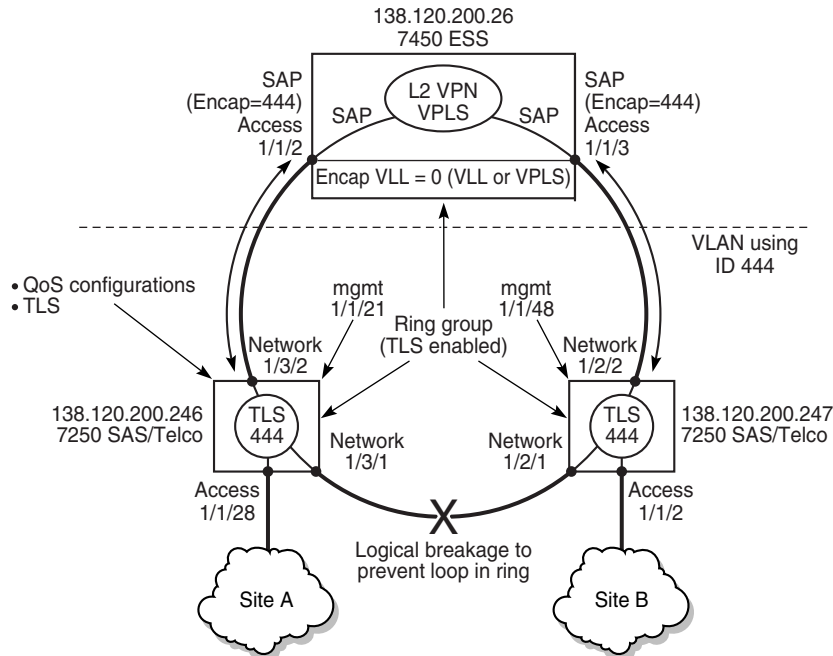
The 5620 SAM supports the creation of VLAN services using Telco devices configured as PE devices. VLANs can be configured as:

- L2 VPN VLAN services
- BTV VLAN services
- Internet access (super-VLAN) services

VLAN ring groups are used to send traffic across an Ethernet ring using copper or fiber optic connections from the source traffic device, for example, from a 7450 ESS to all devices in the ring. Automatic STP configuration on the Telco devices ensures that there is a constant stream of traffic in either direction. Any breaks in the physical links between Telco devices are rerouted.

Figure 18-5 shows a sample VLAN for L2 VPNs.

Figure 18-5 Sample VLAN configuration for L2 VPNs



17676

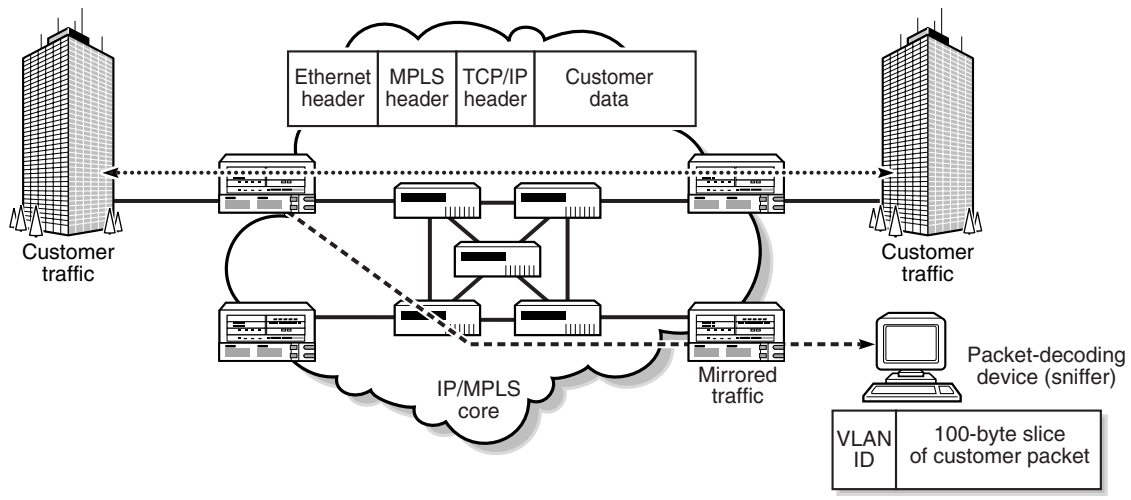
Mirror

The 5620 SAM-O implementation of service mirroring provides mirroring of service traffic packets from any service type.

In a mirror service, originating packets are forwarded to their service destination, and a copy of the entire packet, or a portion of the packet, is sent to the mirror destination. The mirrored packet can be viewed by an operator using a packet sniffer device that is attached to the destination mirror port. A service mirror can have many mirror sources and one destination. Mirroring is performed on a unidirectional basis. The mirrored packets are transported across the core network using IP or MPLS tunneling.

Figure 18-6 shows how service mirroring is performed.

Figure 18-6 Service mirroring



17265

Service mirroring can be used to:

- troubleshoot problems with customer packet delivery and content
- allow service providers to meet regulations by obtaining itemized call records and wiretaps, as authorized by investigative authorities
- simplify the complex analysis networks that are often implemented as overlays to the customer-facing network
- specify full-packet or sliced-packet mirroring, depending on the requirements

Consider the following information before you implement service mirrors.

- The default subscriber is the only subscriber to a service mirror. The default subscriber ID is 1.
- The destination of a service mirror must be a Layer 2 SAP.
- Ingress or egress SAPs, or ingress or egress network ports, can be mirrored. The types of packets that are mirrored differ based on the type of service or port used.
- Service mirror IDs are obtained from the same pool of IDs used by other services. If you manually assign ID values, ensure that you do not assign an ID that belongs to another service.
- Up to 255 mirror destinations are supported per managed network.
- You can use the packet-slicing option to copy a specific packet size for each frame. This option is useful for monitoring network usage without copying the customer data. It also limits the amount of mirrored traffic that streams through the managed devices and the core network.
- If the mirror destination site is not on the same NE as the mirror source, there must be a service tunnel between the source and destination sites.
- The 5620 SAM can automatically create a service tunnel between the source and destination sites in the following case:
 - automatic SDP binding creation is enabled
 - GRE is the transport type
 - no other service tunnel is available between the source and destination sites

- The encapsulation type must be the same for all sites associated with a mirror.
- When the same packet is referenced by multiple mirrors (for example, from a SAP and a port), the packet can only be mirrored once in the following non-configurable order starting with MAC or IP filters:
 - MAC or IP filters
 - MPLS labels (ingress label)
 - SAP
 - port



Caution — Service mirroring can affect performance across the network and in the source and destination devices, and should be planned accordingly.

Optical transport service

An optical transport service is a wavelength that traverses the network between two endpoints, which can be tandem wavelengths in some cases. The path the service takes through the network is defined by the 5620 SAM using shortest path algorithm and the knowledge of the node adjacencies. The path the service takes through an NE is called a cross-connect, or XC. An XC is defined by the ingress and egress points for the service on the NE. The NE physical topology defines the internal path that the service takes through the NE.

The 5620 SAM recognizes the adjoining XCs and managed link XCs as an optical transport service. Each service is assigned a trail identifier and a pair of Wavelength Tracker wave keys. The trail identifier, the ITU channel number (wavelength) and wave key pair are unique in the network. Optical transport services create transport connectivity between router ports and hence they must be created before any IP services.

Services are created by selecting the endpoints, the 5620 SAM is then applied to complete the service creation.

The 5620 SAM supports optical transport services between:

- 1830 PSS
- SR to SR NEs
- alien wavelengths and

Protection schemes:

- diverse route no protection
- protection (Y-cable, OPS, ESNCP), discovery of services
- discovery of Regenerated service

Object life cycle

The object life cycle, or OLC, specifies whether or not excess alarms on a service are displayed in the alarm window. For example, when you configure a service, you can stop the flood of alarms by setting the OLC state of the service to *maintenance* until the service is fully configured. The OLC states are:

- maintenance
- in service

The OLC status can be configured by the user on the following service objects:

- service site
- service
- composite service

You cannot change the OLC state of the following service objects. The OLC state of these objects are affected by the OLC state of the parent objects.

- AccessInterface
 - ServiceAccessPoints
 - L2AccessInterface
 - L3AccessInterface
- SdpBindings
 - MeshSdpBinding
 - MirrorSdpBinding
 - SpokeSdpBinding

If you change the OLC state of an object, the OLC state changes for all objects below it in the containment hierarchy. If an object that is higher in the containment hierarchy is set to a *maintenance* OLC state, you cannot set child objects to an *in service* OLC state. When you change the OLC state of an object, the 5620 SAM-O sends an “[AttributeValueChangeEvent](#)” for the object changed. The 5620 SAM-O does not send this event message for the contained objects that are changed as a result of the parent object OLC state change.

18.2 Workflow to configure a mirror service

The following workflow lists the high-level steps required to create a mirror service. This workflow assumes the following:

- a customer is created.
- the IP or IP/MPLS core network exists.
- any required service tunnels are created.
- access ports, which are the endpoints of the service, are created.
- static routes and/or routing protocols are configured between all routers. See chapter 16.
- the network services the mirror service will use are created.
- a packet-sniffing device at the L2 SAP that is the destination of the mirror service is configured.

- the SAP with the attached packet-sniffing device as the mirror destination is specified.
 - the source of the packets to be mirrored is configured.
- 1 Create the mirror service. See Appendix B for information about how to obtain a sample request to provision a mirror service.
 - 2 As required, view mirror service information.
 - 3 As required, modify mirror service information.
 - 4 As required, delete mirror service information.

18.3 Workflow to configure an optical transport service

The following workflow lists the high-level steps required to configure an optical transport service.

- 1 Specify the customer for the service.
- 2 Specify the A end optical service site.
- 3 Specify the termination point port.
- 4 Specify the Z end optical service site.
- 5 Specify the termination point port.

See Appendix B for information about how to obtain a sample request to provision an optical service.



Note 1 — Optical links have to be established between the 1830 PSS and adjacent non-1830 PSS NEs, and to the other 1830 PSS NEs before an optical service is operational.

Note 2 — A maximum of two sites can be selected for the creation of an optical transport service. These are respectively named an A end and a Z end site by the 5620 SAM.

18.4 Composite services

The 5620 SAM-O XML interface uses the `service.CompositeServiceManager` (comp-svc-mgr) package to manage composite services.



Note — Alcatel-Lucent recommends using the generic package for configuring all objects in the `service.CompositeServiceManager` (comp-svc-mgr) package. See section 14.2 for more information about generic methods. See section 12.2 for more information about the logical object hierarchy.

A composite service is a set of linked services. Composite service functionality supports complex applications that require a combination of services, such as VLAN connections to an HVPLS, an IES spoke into a VPLS, or a VPRN-to-VPLS interconnection.

Services that are owned by different customers can be connected to form a composite service. An example is an HVPLS in which the core VPLS belongs to one customer and the satellite VPLS instances belong to other customers. An HVPLS is considered to be a composite service by the 5620 SAM-O.

Composite services consist of subscriber services, called SCs in the context of a composite service, and connectors. A connector is a bidirectional logical link between two SCs, such as a pair of PW spokes that carry traffic in opposite directions between VLL and VPLS instances, a dot1Q-encapsulated link between a VLAN and a VPLS, or an internal cross-connect.

The term SCP describes a type of connector endpoint. In the case of the services that are available on the 7450 ESS, 7710 SR, 7210 SAS-M, or the 7750 SR, an SCP is a service interface or SAP. For L2 switches, such as the Telco, an SCP may be a network interface, such as an uplink port.

Composite services exist only in the context of the 5620 SAM and are configured through the 5620 SAM GUI or an OSS application. They are unknown to individual network devices. To simplify composite service configuration and to ensure that non-5620 SAM device configuration does not disrupt the management of composite services, the following rules apply to the creation, deletion, modification, and presentation of composite services.

- A composite service can have no SCs.
- A composite service can have zero connectors.
- Two connected SCs can belong to only one composite service.
- A connector between two SCs belongs to only one composite service.
- An SC cannot be removed from a composite service until its connector to the composite service is removed.
- A group of connected services can be moved from one composite service to another.

If a service that is specified for inclusion in a composite service does not currently belong to a composite service, it is added to the composite service regardless of its administrative or operational state. If the specified service is part of an existing composite service, it can be moved to a different composite service. However, SCs that are connected to the specified service are also moved to the new composite service.

Network discovery of composite services

The 5620 SAM-O associates SCs and connectors with composite services during network discovery. The following rules apply to this process.

- When the 5620 SAM-O discovers a valid connector between two services that do not belong to a composite service, a composite service that contains the services and connector is automatically created.
- When the 5620 SAM-O discovers a valid connector between two services and one of the services belongs to a composite service, the other service is added as an SC of the composite service.
- When the 5620 SAM-O discovers a valid connector between two services and the two services are SCs of different composite services, an alarm is raised and the connector is excluded from the 5620 SAM database.

Connector types

There are three types of connectors that join SCs in a composite service:

- SCP-to-SCP
- internal cross-connect
- PW spoke

SCP-to-SCP connectors

SCP-to-SCP connectors can join any two SC types that have service interfaces on the same device or on different devices. A connector between VPLS and VPRN SAPs is an SCP-to-SCP connector, as is a connector between a dot1Q-encapsulated VPLS SAP and L2 switch uplink port of the same encapsulation in a VLAN ring group. The two interfaces must be of the same encapsulation type and encapsulation value. Table 18-1 describes the supported encapsulation types.

Table 18-1 Supported encapsulation types

SAP type	Encapsulation type
Ethernet	Dot1 Q
	Q in Q
	Null
ATM	VPI/VCI
	VPI
FR	DLCI

(1 of 2)

SAP type	Encapsulation type
SONET/SDH	BCP Null
	BCP Dot1 Q
	IPCP
	PPP Auto
	cHDLC
	WAN Mirror
LAG	Null
	Dot1 Q

(2 of 2)

The operational status of an SCP-to-SCP connector depends on the operational status of its endpoints. An alarm raised against one of the endpoints causes an alarm to be raised against the connector. Such alarms are aggregated within the composite service.

Internal cross-connect connectors

An internal cross-connect connector can join any SC types. It uses a CCAG to join two SCs that have SAPs or network interfaces on the same device. This functionality is available in the 7450 ESS, 7710 SR, and 7750 SR. The following rules apply to internal cross-connect connectors.

- A SAP can be connected to another SAP or to a network interface using a CCAG.
- When a SAP or network interface is deleted, the connector associated with it is also deleted.
- The deletion of an internal cross-connect connector causes the associated interfaces and SAPs to be deleted.

The operational state of an internal cross-connect connector depends on the operational state of the CCAG. An alarm raised against the CCAG causes an alarm to be raised against the connector. Such alarms are aggregated within the composite service.

PW spoke connectors

A PW spoke connector generally joins VPLS instances to create an HVPLS service. A PW spoke can, for example, connect IES and VPLS instances to provide distributed Internet access service. The endpoints of a PW spoke connector must be on different devices. PW spoke connectors are subject to restrictions on the SC types that they can join. Table 18-2 shows the SC types that can be linked by PW spoke connectors.

Table 18-2 Valid PW spoke interconnections

SC type	Valid PW spoke SC interconnections
VLL	IES, VPLS

(1 of 2)

SC type	Valid PW spoke SC interconnections
VLAN	—
VPLS	IES, VLL, VPLS
MVPLS	MVPLS
IES	VLL, VPLS
VP RN	—

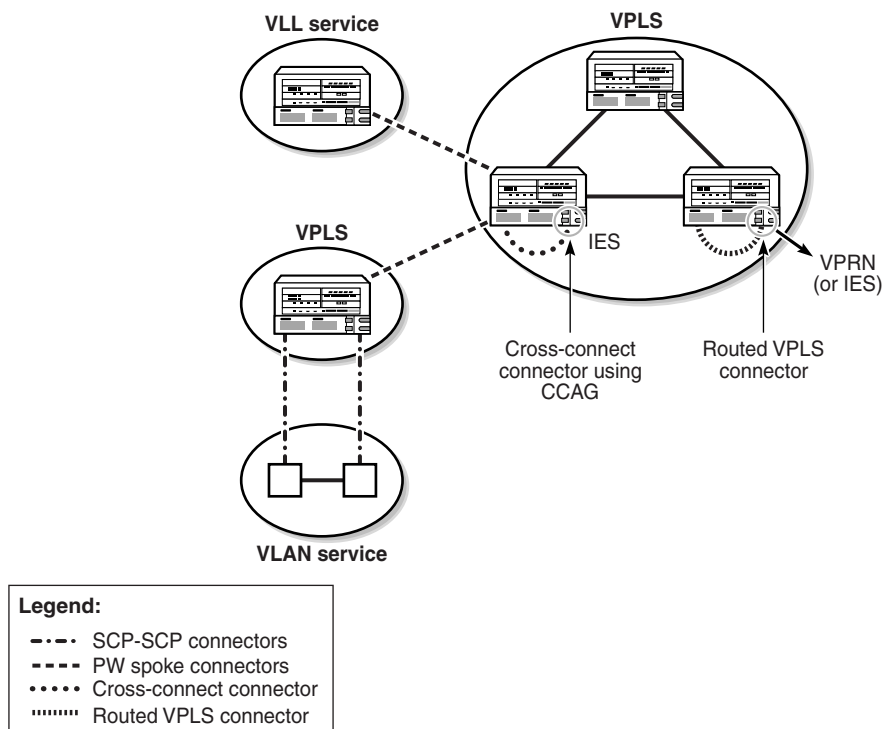
(2 of 2)

The operational state of a PW spoke connector depends on the operational state of the underlying SDP bindings. An alarm raised against one of the SDP bindings causes an alarm to be raised against the connector. Such alarms are aggregated within the composite service.

Sample composite service configuration

Figure 18-7 shows a sample composite service configuration that involves a variety of subscriber services and uses the three SC connector types.

Figure 18-7 Sample composite service configuration



18118

18.5 Workflow to configure a composite service

The following workflow lists the high-level steps required to configure a composite service. See Appendix B for information about how to obtain sample requests to perform these steps.

- 1 Create the composite service.
- 2 Add the service components to the composite service.
- 3 Create a connector between the service components. The 5620 SAM-O supports the following types of connectors:
 - a SCP-to-SCP
 - b PW spoke
 - c Cross connect
 - i Create a CCAG on the router.
 - ii Create the cross connect.

18.6 IGMP snooping

IGMP snooping allows a device to snoop packets sent between IP multicast routers or switches and IP multicast hosts in order to learn the IP multicast group membership. The device checks the IGMP packets for the group registration information, and configures multicasting accordingly.

Without IGMP snooping, multicast traffic is forwarded to all ports, which is the same as broadcast traffic. IGMP snooping ensures that multicast traffic is only forwarded to ports that are members of the specific multicast group, which reduces the amount of multicast traffic passing through the device.

The 7450 ESS and 7750 SR support IGMP snooping for VPLS. A database of group members per VPLS instance is built by listening to IGMP queries and reports from each SAP and SDP of the instance. The reports are forwarded to the multicast routers.

18.7 Workflow to configure IGMP snooping

The following workflow lists the high-level steps required to configure IGMP snooping. See Appendix B for information about how to obtain sample requests to perform these steps.

- 1 Create a VPLS service.
- 2 Enable IGMP snooping from the site of the VPLS.
- 3 As required, configure IGMP snooping parameters from the site of the VPLS.

- 4 As required, configure IGMP snooping parameters from the spoke SDP.
- 5 As required, configure MFib parameters from the site.

18.8 DHCP relay configuration

DHCP relay interconnects DHCP clients with a DHCP server connected to another LAN segment or network. A DHCP relay agent listens for transmissions from DHCP clients and relays them to the DHCP server. The server responds back to the agent, which then sends the server response back to the client.

5620 SAM supports DHCP relay on VPLS SAPs, VPRN/IES SAPs, spoke/mesh SDP circuits.

18.9 Workflow to configure DHCP relay

The following workflow lists the high-level steps required to configure DHCP relay. See Appendix B for information about how to obtain sample requests to perform these steps.

- 1 Configure DHCP Relay on a VPLS SAP.
 - i Create a VPLS service.
 - ii On the VPLS L2 interface, configure DHCP relay.
- 2 Configure DHCP Relay on a VPRN SAP.
 - i Create a VPRN service.
 - ii On the VPRN L3 interface, configure DHCP relay.
- 3 Configure DHCP Relay on a SDP circuit.
 - i Create a VPLS service.
 - ii On the VPLS L2 interface SDP circuit, configure DHCP relay.

18.10 IPsec management

The 5620 SAM IPsec VPRN enables the secure extension of a corporate VPN over uncontrolled or untrusted private and public networks. The ISA-IPSEC MDA on the 7750 SR, Release 6.1 or later, provides IPsec tunneling and encryption between sites.

You can use the 5620 SAM to configure IPsec sessions and the security associations that are required in a bidirectional IPsec tunnel. You can configure multiple IPsec tunnels for each VPRN.

IPsec VPRN services include IPsec tunnels that terminate on IES or VPRN IPsec gateways, and support Layer 3 forwarding through an interface connected to an IPsec tunnel.

Typical IPsec configuration example

The following is a typical IPsec configuration example.

- Individual hosts connect over the Internet into an IES or VPRN service IP interface. The public IP address is local-gateway-address.
- The IES or VPRN service interface defines the public subnet for the secure tunnel and is associated with the ISA by the IPsec group.
- An IPsec interface is configured on a VPRN.
- Tunnel security profiles can be configured per IPsec interface.

See the *5620 SAM User Guide* for more information about IPsec management.

You can use the 5620 SAM-O to configure VPRN services to which individual hosts connect over the Internet on to an IES or VPRN IPsec gateway. You can configure individual or multiple IPsec interfaces for each VPRN service. Multiple tunnel security profiles can be configured for each IPsec interface.

5620 SAM-O IPsec configuration

The 5620 SAM-O XML interface uses the IPsec package to enable secure exchange of packets at the IP layer. This package allows you to create:

- IPsec endpoints on L3 access interfaces on IES and VPRN services
- IPsec interfaces
- IPsec tunnels
- IPsec gateways

You can also use the 5620 SAM-O to create the association between IPsec components, public and private services, to for a secured VPN. When creating secure or delivery services for the IPsec VPN, you need to specify the access interface for the secure service, and an access interface for the delivery service.

OSSI configuration requirements for static tunnel type

For creating static tunnels, the xml script must include parameters of an IPsec tunnel under the IPsec interface.

OSSI requirements for dynamic tunnel type

For creating dynamic tunnels, the xml script must include parameters of an IPsec gateway under the L3 access interface.

The OSSI requires that you specify all of the necessary objects and child objects you need to set up the desired tunnel.

18.11 Workflow to configure IPsec — option A

The following workflow lists the high-level steps required to configure IPsec. See Appendix B for information about how to obtain a sample for creating the IPsec components in the following workflow.

- 1 Provision the ISA-IPSEC MDA on the 7750 SR, Release 6.1 or later.
 - i Create or configure ISA-IPsec groups.
 - ii Assign the active and backup IPsec group members to the ISA-IPsec groups.
- 2 Configure security policies.
- 3 Configure IPsec tunnel templates.
- 4 Create a VPRN.
 - i Configure the IPsec security policy.
 - ii Configure the IPsec security policy entries.
- 5 Create the private-facing IPsec interface on the VPRN.
 - i Create a private IPsec SAP.
 - ii Configure ingress and egress policies.
- 6 Create the public side of the IPsec service.
 - i Create an L3 access interface on an IES or VPRN.
 - ii Define the IPsec public SAP for the L3 access interface.
 - iii Specify the IPsec gateway, if required.
- 7 Create IPsec tunnels on the VPRN IPsec interface.
 - i Create one or more IPsec tunnels.
 - ii Configure the manual or dynamic keying.
- 8 Configure the static route.

18.12 Workflow to configure IPsec — option B

The following workflow lists the high-level steps required to configure IPsec. See Appendix B for information about how to obtain a sample for creating the IPsec components in the following workflow.

- 1 Create an IPsec VPN.
- 2 Select the service sites for an IPsec VPN.
- 3 Create or select the secure service for an IPsec VPN.
- 4 Create or select the delivery service for an IPsec VPN.

- 5 Select the IPsec group for an IPsec VPN and fill out the required IP addresses for the tunnel type.
- 6 Select all required policies and fill out the required information for the tunnel type.

18.13 Customer and residential subscriber configuration overview

Customer configuration

Customers are the service providers who pay for services from a network provider. The 5620 SAM-O allows you to create and manage customers of services by using the `subscr` package.

An end user is the recipient of application content that is delivered through a service. The service is a means of transport for the application content and is owned by a service subscriber. For example, an end user subscribes to an high-speed Internet access service, and the Internet access service is owned by a service provider who is a service customer of the network provider.

See the *5620 SAM User Guide* for more information about customer management.

OSS applications can use the `subscr.Subscriber` class to develop an XML request that will create and configure a customer. The XML request can include a `subscriberId` and other contact information such as `subscriberName`, address, etc. See the `subscr` package in the *5620 SAM-O XML Reference* for more information.

Each customer is associated with a subscriber ID. The basic required entity is the subscriber ID value, which is assigned when the customer account is created. The subscriber ID can be used to modify customer information. When you associate a customer with a service, you can use the `subscriberID` or the `subscriberName`.

Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the Advanced SDK - Configuration - Subscribers package. This package has samples for creating, modifying, and deleting customers.

Residential subscriber

The 5620 SAM-O uses the `ressubscr` package to create and manage residential subscribers.

A residential subscriber is an end recipient of multiple service offering, such as VoD, VoIP, or other triple play applications. A residential subscriber host, sometimes called a subscriber host or simply a host, is an end device, such as a computer or set-top box that connects to the provider network and receives the service traffic.

In the context of the 5620 SAM-O, a residential subscriber, sometimes called a subscriber, has a unique identifier that associates a group of end-user devices with policies. A subscriber can be associated with multiple SAPs on multiple NEs, and a customer can be associated with multiple subscribers.

Configuring residential subscriber management on the 5620 SAM involves the creation and configuration of the following components, depending on the service delivery model:

- subscriber identification policy
- subscriber profile
- SLA profile
- subscriber explicit map (optional)
- ANCP policy
- PPPoE policy
- MSAP policy

A host requires subscriber-profile and SLA-profile associations to gain access to the network. Profiles define service attributes for hosts such as scheduling, accounting, security, and traffic prioritization by application type. A profile uses existing 5620 SAM policy definitions and allows the customization of policy parameters using override values.

See the *5620 SAM User Guide* for more information about residential subscriber management.

18.14 Workflow to configure and manage residential subscribers

The following workflow provides the steps required to configure some of the components required for residential subscribers. Each of the steps represents an XML request that can be developed for the particular operation.

Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the Advanced SDK - Configuration - Subscribers - ResidentialSubscriber package for XML scripts for Creating SLA profile, Create SubscriberId Policy and Create Subscriber Policy below.

- 1 The service that effectively delivers the proposed service offerings.
- 2 The following policies:
 - Policies such as Ingress and Egress scheduler policies for the subscriber hosts in accordance with the customer SLA.
 - An accounting policy for the customer.
 - Access ingress and access egress QoS policies for the different applications, for instance, HSI and VoIP, and levels of service, for example gold, silver, and bronze, that subscriber hosts are to receive. Ensure that the queues defined in the QoS policies reference the previously created scheduler policies.
- 3 Create a unique subscriber identification string for the subscriber.
- 4 Create SLA profiles that name the previously created QoS policies.
 - Create a different SLA profile for each class of service offering.
 - Use override values to customize the policy values, as required.

- 5 Create a subscriber profile.
 - Choose the previously created ingress and egress schedulers.
 - Choose the previously created accounting policy.
 - Enable accounting.
 - Associate one or more of the previously created SLA profiles with the subscriber profile.
- 6 Create a subscriber identification policy.
- 7 Create a subscriber explicit map, if required.
- 8 Supply the customer with the subscriber identification string and any IP information that is required for provisioning the residential subscriber hosts.
- 9 Assign the SLA profile, subscriber profile, and subscriber identification profile to the service, if required.
- 10 Turn up the service after the subscriber hosts are provisioned.

19 – Script and template configuration management

- 19.1 Overview 19-2**
- 19.2 Script management 19-2**
- 19.3 Workflow to execute a script 19-3**
- 19.4 XML API template configuration management 19-3**

19.1 Overview

The script management and template management components of the 5620 SAM allow users to develop, manage and execute customized provisioning actions to simplify the operation of their 5620 SAM network.

19.2 Script management

Script management is typically delivered by the 5620 SAM GUI which provides the framework to develop scripts to create, delete, configure, or view objects in the 5620 SAM network. See the *5620 SAM Scripts and Templates Developer Guide* for more information about script and template management. An OSS client can use the 5620 SAM script manager for the following:

- invoking XML scripts outside of the 5620 SAM GUI framework
- XML script and OSS software troubleshooting

The 5620 SAM-O uses the script package to allow for script management. There are two types of scripts:

- CLI script—used to securely create and manage scripts executed against managed devices, including generic network elements
- XmlAPI script—is the same type of script used for OSS but are typically managed within 5620 SAM, allowing them to be used from the 5620 SAM GUI, and perform complex actions on not just network elements, but 5620 SAM itself, if required.



Warning — CLI and XML API scripts that are not correctly applied or created can cause serious damage to the network. Alcatel-Lucent recommends that system administrators clearly define user responsibilities for script usage, and ensure that scripts are verified and validated before they are executed on devices in a live network.

See the appropriate device hardware documentation for the applicable CLI commands, structure, and usage.



Note — The generic methods are not used to configure scripts, instead use the various configuration methods that are included for the various classes in the script package. See the *5620 SAM-O XML Reference* for more information.

19.3 Workflow to execute a script

The following workflow outlines the high-level steps necessary to create either CLI or XML API scripts. Each of the steps provides guidance on the type of XML requests required to perform a particular configuration action. Contact your Alcatel-Lucent OIPS technical support representative for sample XML scripts from the advanced SDK configuration script management package for each step.

- 1 Create script object using the `script.Scriptmanager` class.
 - a Create CLI script object.
 - b Create XMLAPI script object.
- 2 Create script version.
 - a For CLI script version, include the CLI commands for the target nodes.
 - b For XML API script version, include velocity content and configuration details.
- 3 Create script target on instance
 - a For CLI, create a script target that identifies the node and specific node parameter targets.
 - b For XML API, create a script instance for an XMLAPI script that identifies the `targetObjectFullName` of the object and parameters to be configured.
- 4 Preview scripts.
- 5 Start and stop the execution of scripts.
- 6 Retrieve script results of executions using FTP or SOAP/XML.

You can synchronously or asynchronously execute the scripts. For synchronous execution, a response attribute identifies the success or failure of script execution. For asynchronous execution, the OSS client can either wait for a JMS notification or poll for status on the execution of the script.

The script results are stored in a file for the OSS client to collect. When the OSS client retrieves the results using the SOAP interface, the result is stripped of non-printable characters embedded in the response.

19.4 XML API template configuration management

A template is an object managed by the 5620 SAM that is used as a starting point for configuring complex managed objects such as services and tunnels. The 5620 SAM supports the configuration of templates using XML API script-based templates for services, tunnels and some equipment. Templates also use the 5620 SAM GUI framework to create, modify, or delete template-able objects that are defined in the *5620 SAM Scripts and Templates Developer Guide*.

Templates use the 5620 SAM GUI framework for configuring services, tunnels and equipment as an alternative to OSS requests, even though the same XmlAPI commands are used.

Integrated products

20 — 5650 CPAM integration configuration

20 – 5650 CPAM integration configuration

20.1 5650 CPAM integration overview 20-2

20.2 5650 CPAM OSS interface overview 20-4

20.3 5650 CPAM OSS packages 20-11

20.1 5650 CPAM integration overview

The 5620 SAM can be configured to interwork with the 5650 Control Plane Assurance Manager. The 5650 CPAM provides real-time control IGP topology capture, inspection, visualization, and troubleshooting. The integration with the 5620 SAM allows the 5650 CPAM to cohesively tie routing layer information to infrastructure, such as network routes, service tunnels, LSPs, edge-to-edge service layer connectivity, and rich OAM test management.



Note — You require a 5650 CPAM license key to install the 5650 CPAM.

See the *5650 CPAM User Guide* for more information about the 5650 CPAM and licensing information.

The 5650 CPAM and 5620 SAM integration supports the following operational activities:

- **network planning**
Planning activities are optimized with real-time topology and strong linkages between services and infrastructure layers in the 5620 SAM GUI and 5620 SAM-O OSS interfaces.
- **network operations**
Real-time topology and multi-layer highlighting allows you to rapidly assess the state of services, tunnels, and routing on the IGP and IP/MPLS maps.
- **network troubleshooting**
Historical OAM trace, SPF and RSVP path, and checkpoints allow you to rapidly detect and resolve service level issues whose root cause is in the IP or MPLS layers.
- **network restoration**
Checkpoints and real-time views of IP/MPLS and service and tunnel infrastructure allow you to restore and plan networks.
- **proactive assurance**
5650 CPAM alarms, network route and tunnel inspection lists, validation functions, checkpoints, and multi-layer views allow you to detect routing faults.

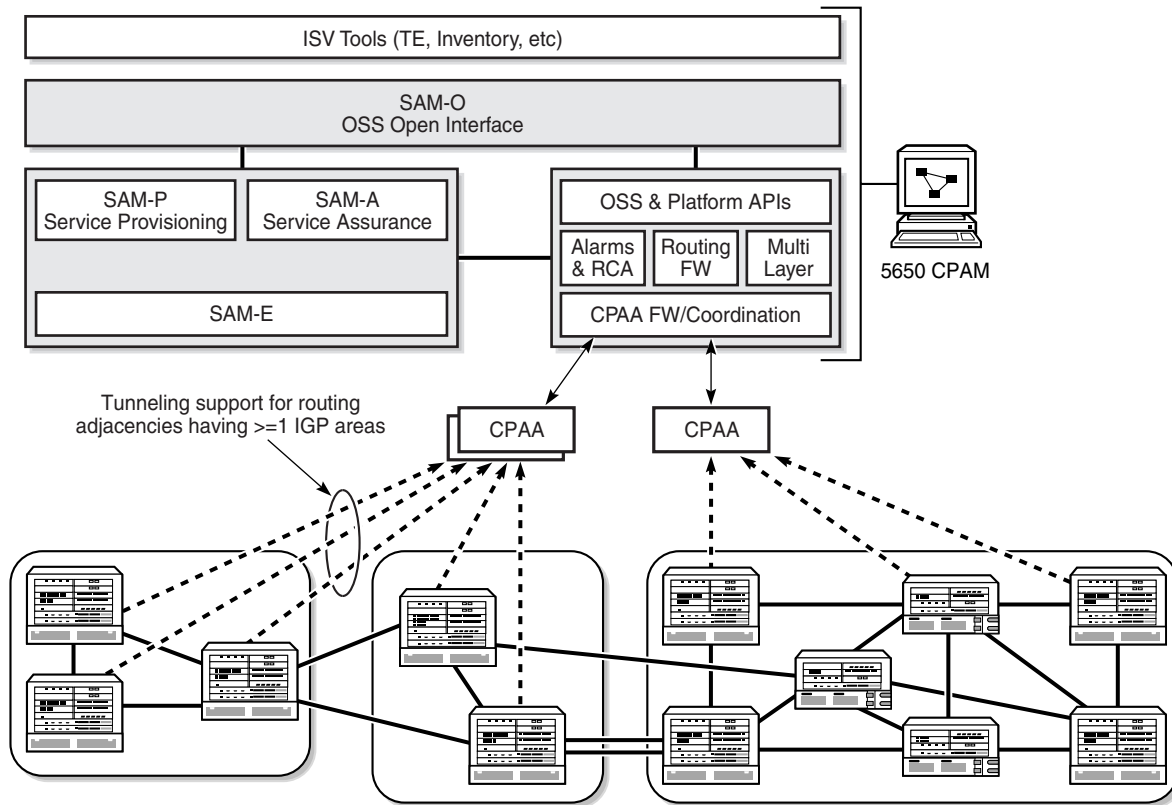
The 5650 CPAM provides a tight eastbound and westbound integration with the 5620 SAM. This integration allows for a real-time view of the network including routing topology and associated configurations whether performed on the GUI, OSS interface, or by CLI. The 5650 CPAM can leverage 5620 SAM redundancy and offer tight navigation between protocol maps and 5620 SAM-managed objects, such as protocol links. In addition, the 5620 SAM GUI supports the management of the 7701 CPAA platform.

When the 5650 CPAM and the 5620 SAM share the same database, the 5650 CPAM can access objects managed by 5620 SAM and display them on the 5650 CPAM topology views. Without integration with the 5620 SAM, the following functions are not available:

- service, LSP, multicast, and OAM highlights
- historical LSP active paths
- LDP and RSVP interface display on the MPLS view

Figure 20-1 shows the 5650 CPAM architecture. See the 5650 CPAM documentation suite for information about how to configure application interoperation.

Figure 20-1 5650 CPAM and 5620 SAM interworking



19083

There are two main components to the control plane assurance management solution:

- a server component, the 5650 CPAM route controller
- a route analyzer component, the 7701 CPAA

The 5650 CPAM server component contains 7701 CPAA control frameworks, applications, and coordination functions for the distributed 7701 CPAAs and provides the applications that are required to leverage the data provided by the 7701 CPAA platform.

The 5650 CPAM route controller, or server, communicates with several GUI and OSS/I clients using the same API as the 5620 SAM. When the 5650 CPAM route controller is integrated with a 5620 SAM, the 5620 SAM and 5650 CPAM versions must be compatible. In addition, the 5650 CPAM route controller can run independently with one or multiple 5620 SAM servers.

The 7701 CPAA is a mountable rack that provides an analysis and distributed computing platform. This platform uses a passive version of the industry-tested and -proven 7750 SR routing code base to guarantee a high degree of interoperability with customer networks. The 7701 CPAA acts as a special-purpose routing element whose role is to passively peer with the network.

The 7701 CPAA is supported by the 5620 SAM as a special-purpose NE.

20.2 5650 CPAM OSS interface overview

The 5620 SAM-O OSS interface module includes 5650 CPAM packages and methods that provide a smooth integration path for the export of real-time protocol or IP-level topology to advanced OSS applications, such as TE tools.

The following 5650 CPAM features are of interest to OSS developers:

- multiple topologies
- administrative domains
- checkpoints and real-time topology changes
- route management
- path monitoring
- fault management

Multiple topologies

The 5650 CPAM allows OSS applications to retrieve information about the following network protocol topologies:

- OSPF
- ISIS
- IGP
- BGP
- MPLS — MPLS and LDP information is collected from the 5620 SAM

The 5650 CPAM topology package is used to manage routing topology objects and routing alarms. The 5650 CPAM supports ISIS, OSPF, and BGP.

Both the OSPF and ISIS topologies include information about routers, subnet objects, links, OSPF areas, ISIS levels, and LSDB updates.

The IGP topology includes information about routers, protocols for both OSPF and ISIS, and information about IGP Links.

The MPLS topology includes information about IGP links, MPLS, RSVP, and LDP interfaces for routers.

The BGP topology includes BGP route information originating from a 7701 CPAA that monitors a BGP AS and maintains a BGP RIB. The 7701 CPAA, Release 2.0 and later, supports the following BGP configurations:

- BGP AS
- BGP confederation AS
- BGP sub-AS

5650 CPAM administrative domains

An administrative domain represents a logical routed network. Administrative domains are configured by the operator to reflect the logical structure of the network.

The following types of administrative domains are supported:

- **IGP Administrative Domain**

Represents an IGP routed network. In an IGP administrative domain, there can be a single OSPF route network and a single ISIS routed network. An IGP Administrative Domain may have both OSPF and ISIS at the same time. For historical reasons, this is modeled as the topology.AutonomousSystem.

- **BGP Autonomous System**

Represents either a standard BGP AS, a BGP confederation AS, or a BGP sub-AS (confederation member). BGP sub-ASs are configured as children of a confederation BGP AS. Each standard BGP AS (or sub-AS) may be associated with a single IGP administrative domain. The same IGP administrative domain may be assigned to multiple BGP AS (or sub-AS). Note that a confederation BGP AS cannot be assigned to an IGP administration domain. The class is topology.BgpAutonomousSystem.

You can specify the role of a 7701 CPAA, Release 2.0 and later: IGP, BGP, or both. The role of the 7701 CPAA indicates to the 5650 CPAM whether information from a protocol should be interpreted. A 7701 CPAA with an IGP role must be assigned to an IGP administrative domain. A 7701 CPAA with a BGP role must be assigned to a BGP AS.



Note — See the *5650 CPAM User Guide* for more information about the supported topologies.

Topology checkpoints

A topology checkpoint is a snapshot of some part of the network (specified by rules in the subclass) at some specific time. When you create a checkpointed object using a real network object, all of the properties of the real object—for example, metric and bandwidth on IGP links—are copied to the checkpointed object.

After you have set up your network and the network is operational, you can checkpoint the network to create a snapshot of the current state—which can include routers, links, metric configuration, or bandwidth usage—and compare it with checkpoints collected at different times.

All checkpoints exist within an IGP admin domain, this means that you can only create checkpoints on OSPF and ISIS routing protocols.

OSPF checkpoints

You can create OSPF checkpoints for an OSPF area within an IGP administrative domain. Each OSPF area can have multiple checkpoints separated by time.

The following OSPF objects are included in an OSPF checkpoint:

- OSPF area
 - backbone
 - standard
 - stub
 - total stub
 - NSSA no type 5
 - NSSA no summaries
- all OSPF area links (point to point and broadcast)
- all OSPF routers
 - ABR
 - ASBR
 - 5620 SAM-managed
 - third-party managed
- OSPF area subnets
- 7701 CPAA

ISIS checkpoints

You can create ISIS checkpoints in each ISIS routing domain within an IGP administrative domain. Each ISIS routing domain can have multiple checkpoints, separated by time.

The following ISIS objects are included in an ISIS checkpoint:

- ISIS routing domain
 - Level 1—all Level 1 routers that are connected in a single domain within an IGP administrative domain. All Level 1 routers are discovered by one 7701 CPAA.
 - Level 2—all Level 2 routers within an IGP administrative domain
- all ISIS routers
 - L1
 - L2
 - L1/L2
 - ASBR
 - 5620 SAM-managed
 - third-party managed
- ISIS routing domain subnets
- 7701 CPAA

Checkpoints using OSS

When you gather topology information via an OSS, you must use a filter to separate checkpoint data from real-time data.

For real-time:

```
<equal name="isCheckpointObject" value="false"/>
```

For checkpoint:

```
<equal name="isCheckpointObject" value="true"/>
```

You can use the 5650 CPAM to diagnose problems in your network by comparing two checkpoints of the network and viewing the details about the differences in configuration and topology changes.

When you compare two checkpoints, you can specify whether the comparison is of checkpointed areas, routers, links, or subnets, or a combination of these objects.

The first checkpoint that you select is the base for the comparison. You can swap the order of the checkpoints so that the second checkpoint that you select becomes the base for the comparison. You can specify a filter to limit what comparison results are visible.

The objects involved in an OSPF checkpoint comparison include the following:

- OSPF area
- all OSPF area links
- all OSPF routers
- OSPF area subnets

The objects involved in an ISIS checkpoint comparison include the following:

- ISIS routing domain
- all ISIS routing domain links
- all ISIS routers
- ISIS routing domain subnets

Code 20-1 provides an example on how to do checkpoint comparison:

Code 20-1: Checkpoint comparison sample

```
<SOAP:Body>
  <generic.GenericObject.compareObjects xmlns="xmlapi_1.0">
    <comparisonFilter>
      <generic.ComparisonFilter>
        <classesToProcess>
          <!-- Insert classes to compare. Such as: -->
          <!-- <string>topology.OspfLink</string> -->
        </classesToProcess>
        <includeOnlyDiffs>true</includeOnlyDiffs>
      </generic.ComparisonFilter>
    </comparisonFilter>
    <!-- FDN of first checkpoint -->
    <base>tpgy-mgr:name-MyNetwork-AS-0:checkpoint-8</base>
    <!-- FDN of second checkpoint -->
    <compareTo>tpgy-mgr:name-MyNetwork-AS-0:checkpoint-2</compareTo>
  </generic.GenericObject.compareObjects>
</SOAP:Body>
```

Checkpoint schedule policies

You can use the 5650 CPAM to schedule checkpoints by creating checkpoint schedule policies. The 5650 CPAM scheduling functionality allows the creation of 5620 SAM-based schedules for the automatic execution of checkpoint tasks at designated times.

You can associate a schedule that you create using the 5650 CPAM with a checkpoint scheduled task that can be immediately processed, scheduled for later execution, or retained for future use. A scheduled task must be created in the checkpoint scheduler policy configuration form for the scheduled task. Once scheduled tasks are created they are associated with a schedule. A schedule is configurable for one-time or ongoing task execution. You can optionally specify the time at which an ongoing schedule is to stop functioning.

The following are the topology checkpoint classes for scheduling checkpoints and defining schedule policies:

- **CheckpointScheduledTask**
Represents a specialized Checkpoint Scheduled Task used for performing scheduled checkpoints.
- **CpSchedulePolicy**
Defines the policy for creating scheduled checkpoints, sub-classes include topology.IsisCpSchedulePolicy and topology.OspfCpSchedulePolicy.

Route management

A managed route is an IP path from a given source router to a given destination router. An IP path represents one or more GRE or LDP tunnels, and loose-path RSVP LSPs, of the same source and destination. Managed routes can span multiple areas and, therefore, multiple 7701 CPAAs.

The route manager object can be used by an OSS application to store multiple managed routes. The OSS is then informed, via the OSS I event channel, of the changes to the paths of the managed routes. A separate route manager must be created for each application. This allows for separate sets of managed routes for different applications. JMS events may be filtered by application.

The OSS application is responsible for creating the route manager using the regular generic.GenericObject.configureChildInstance method using the topology.TopologyManager singleton as the parent.

The route manager and its set of managed routes are persisted. The paths of the managed route are not persisted.

Route management sample workflow

The following workflow provides sample OSS XML requests for performing route management tasks.

- 1 Create a managed route client, as shown in Code [20-2](#).
- 2 Create a request to add or register managed routes, as shown in Code [20-3](#).
- 3 Monitor managed routes, as shown in Code [20-4](#).

Code 20-2: Create managed route client

```

<SOAP:Body>
  <generic.GenericObject.configureChildInstance
xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <distinguishedName>tpgy-mgr</distinguishedName>
    <childConfigInfo>
      <topology.RouteManager>
        <actionMask><bit>create</bit></actionMask>
        <applicationName>TestRoutes</applicationName>
      </topology.RouteManager>
    </childConfigInfo>
  </generic.GenericObject.configureChildInstance>
</SOAP:Body>

```

Code 20-3: Request to add or register managed routes

```

<SOAP:Body>
  <topology.RouteManager.registerRoutes xmlns="xmlapi_1.0">
    <deployer>immediate</deployer>
    <!-- FDN of Managed Route Client -->
    <instanceFullName>tpgy-mgr:application-TestRoutes</instanceFullName>
    <routeKeySet>
      <topology.RouteKey>
        <sourceType>ipv4</sourceType>
        <source>10.1.202.167</source>
        <sourceLen>32</sourceLen>
        <destType>ipv4</destType>
        <dest>10.1.202.10</dest>
        <destLen>32</destLen>
      </topology.RouteKey>
    </routeKeySet>
  </topology.RouteManager.registerRoutes>
</SOAP:Body>

```

Code 20-4: Monitor managed routes

```

<SOAP:Body>
  <topology.RouteManager.retrieveRoutes xmlns="xmlapi_1.0">
    <!-- FDN of Managed Route Client -->
    <instanceFullName>tpgy-mgr:application-TestRoutes</instanceFullName>
    <routeKeySet>
      <topology.RouteKey>
        <sourceType>ipv4</sourceType>
        <source>10.1.202.167</source>
        <sourceLen>32</sourceLen>
        <destType>ipv4</destType>
        <dest>10.1.202.10</dest>
        <destLen>32</destLen>
      </topology.RouteKey>
    </routeKeySet>
  </topology.RouteManager.retrieveRoutes>
</SOAP:Body>

```

Path monitoring

OSS applications that are integrated with the 5650 CPAM can be used to monitor a limited set of network routes—the route between any two routers seen by the 5650 CPAM. When a network topology changes, such as a link metric or state change, the system evaluates whether the routes of any registered path are affected. If this is the case, new routes are recorded and the OSS clients are immediately informed. If there is no route for a monitored path as a result of a topology change, a record is logged.

The 5650 CPAM supports the following types of path monitoring:

- **IP network path monitoring**

Monitors the IP forwarding path between two system IP addresses. Both unidirectional and bidirectional IP path monitoring is supported within a single IGP administrative domain. Bidirectional IP paths monitor path divergence and the 5650 CPAM raises a self-clearing alarm when the two paths do not share the same set of IP links.

- **LSP path monitoring**

Monitors the path of a given LSP. RSVP LSP path monitoring is similar to network route monitoring.

The 5650 CPAM monpath package is used to monitor changes to the paths of IP routes and LSPs. The changes to the path are recorded as they occur by 5650 CPAM IP and LSP path monitors.

The createMonitoredPath method is used to create multiple monitored paths from the passed in LSPs or service tunnels. If a monitored path is not created based on the *fullname* passed into the createMonitoredPath method, the LSP or service tunnel is ignored.

You can use the 5650 CPAM to determine the cause event of an IP or LSP path record.

Fault management

The fault management system that the 5650 CPAM-O provides include:

- RCA audit policy for identifying problems or errors in protocol-related and control plane configuration
- threshold reaching alarms generated by the 7701 CPAA

RCA audit policies

The 5650 CPAM allows you to perform on-demand verifications of the IP/MPLS configuration of IS-IS and OSPF routing protocols on 5620 SAM-managed NEs. RCA audit policies identify problems or errors in protocol-related and control plane configurations. These configuration problems, if not discovered in a timely manner, can have significant effects on higher layers, such as MPLS and VPN.



Note — The OSS user must be assigned the Administrator or RCA Verification scope of command role in order to create, modify, and execute all RCA audit policies. The results of an RCA audit are associated with the object, not the user.

You can create RCA audit policies for IGP administrative domains to verify the configuration of network objects on 5620 SAM-managed routers. For each audit policy that you create, you can specify one or more policy entries that define the scope of configuration that is verified by the audit and the severity of the related problem.

For example, an RCA audit policy on an OSPF interface verifies the configuration of several default attributes, such as the configured MTU and Hello interval. A problem with the specified severity is raised against the IGP administration domain for IS-IS misconfigurations, the OSPF area for OSPF misconfigurations, and the IS-IS or OSPF interface for interface misconfigurations, and so on, depending on the policy entry. The results of the verification are displayed in two components trees—the problem tree and the object tree. You can expand the problem tree to view the problem type and other objects affected by the same problem.

The 5650 CPAM rca package is used for policy-driven IP/MPLS configuration audits for OSPF and ISIS routing protocols.

The following workflow provides sample workflow for performing RCA audit policy tasks.

- 1 Create an RCA audit policy. Configure the attributes that are verified for each RCA audit policy entry.
- 2 Run the RCA audit policy on a specific object.
- 3 Identify the problems and correct the configuration.
- 4 Repeat steps 2 and 3, as required.

Threshold reaching alarms

In addition to routing alarms generated by routers, there are threshold reaching alarms generated by the 7701 CPAA. The alarm generation process uses the routing data collected by the 7701 CPAA. The generated alarms are sent to the 5650 CPAM route controller using a proprietary protocol over the TCP channel. The alarms are then available to 5650 CPAM OSSI clients via JMS.

All 5650 CPAM-related messages are part of the CPAM category, for example `ALA_category = CPAM`.

The following are not included in the CPAM category:

- Configuration changes
 - interfaces
 - protocol
 - hardware (cards, daughter cards, ports)
 - MPLS
 - LDP

See the *5650 CPAM User Guide* for more information about the 5650 CPAM threshold reaching alarms for BGP, ISIS and OSPF.

20.3 5650 CPAM OSS packages

Table 20-1 describes the packages that include the 5650 CPAM objects.

Table 20-1 5650 CPAM packages

Package	Description	Dependencies
monpath	For configuring monitored paths on the network	—
rca	For 5650 CPAM policy-driven IP/MPLS configuration audits for OSPF and ISIS routing protocols	—
topology	For routing topology object configuration and management, such as: <ul style="list-style-type: none">• checkpoints• alarm thresholds• administrative domains• autonomous systems	—
topologysim	For simulated routing topology object configuration and management, such as: <ul style="list-style-type: none">• monitored paths• routers	—

Appendices

- A. Troubleshooting
- B. 5620 SAM-O SDK library of samples

A. *Troubleshooting*

A.1 Troubleshooting client OSS application problems A-2

A.1 Troubleshooting client OSS application problems

The following procedures describe how to troubleshoot OSS application-specific problems.



Note — See the *5620 SAM Troubleshooting Guide* for more information about how to troubleshoot common GUI and OSS client application issues.

Table A-1 Troubleshooting client OSS application problems procedures

Procedure
The client OSS application cannot communicate with the server
An attempt to log in to the 5620 SAM server fails
Receive insufficient privileges to perform this operation on an object exception when performing an action on a 5620 SAM object
An XML request fails
Unable to perform an action using the 5620 SAM-O
Identifying XML messages from specific users
Receive a java.lang.UnsupportedClassVersionError when sending scripts using the PostXML tool
Receive a java.net.ConnectException when sending scripts using the PostXML tool
Receive a java.net.ConnectException when sending HTTP requests to the 5620 SAM-O server
The OSS client cannot connect with the HTTP or JMS server
The OSS client cannot perform find or findToFile requests

Procedure A-1 The client OSS application cannot communicate with the server

The following XML API ping sample can be used to test whether the OSS application can access the 5620 SAM server. Information in the SOAP/XML request includes:

- standard SOAP user and password encoding
- the ping command to look for the release of XML on the server, which in this case, is Release 1.0

An exception response to a failed ping may result in many types of return messages, for example:

- socket timeouts
- HTTP 404 errors
- connection exceptions

1 Run a ping command similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP:Envelope>
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <security>
        <user>user name</user>
        <password>MD5-hashed user password</password>
      </security>
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <ping xmlns="xmlapi_1.0"/>
  </SOAP:Body>
</SOAP:Envelope>
```

2 Review the three-digit HTTP response code.

- 200 to 299—indicates success, and the problem is not with communication to the server
- 400 to 499—indicates that the error is on the client side, and the request contains bad XML syntax, or cannot be fulfilled
- 500 to 599—indicates that the error is on the server side, and the request is not the problem

3 Check the XML API response for any indication of other communication problems. The following is a successful response message to the ping. The response indicates the correct release 1.0 of XML on the server that responds to the ping.

```
<SOAP:Envelope>
  <SOAP:Header>
    <header xmlns="xmlapi_1.0">
      <requestID>clientName@requestId</requestID>
    </header>
  </SOAP:Header>
  <SOAP:Body>
    <pingResponse xmlns="xmlapi_1.0"/>
  </SOAP:Body>
</SOAP:Envelope>
```

Procedure A-2 An attempt to log in to the 5620 SAM server fails

You receive the following SOAP exception message if you enter an incorrect username or password, if the username or password is missing, or if the password is not hashed:

```
<SOAP:Fault>
<faultcode>SOAP:Client</faultcode>
<faultstring>[security] Login failure.</faultstring>
<faultactor>XmlApi</faultactor>
- <detail>
<requestID>XmlApiClient:0</requestID>
</detail>
```

```
</SOAP:Fault>
```

For more information about MD5-hashed passwords, see chapter 3.

Procedure A-3 Receive insufficient privileges to perform this operation on an object exception when performing an action on a 5620 SAM object

You receive the following exception if you are not assigned the appropriate span of control for an object:

```
[ app: generic ] [ class: GenericObject ] [ instance: instance:object ] [ cause: Method Invocation Failed (13) ] [ descr: Insufficient privileges to perform this operation. User does not have create-update-delete access to class classname ] [ nested exception: null ]
```

Check with your administrator to determine whether you are assigned the appropriate span of control to allow you to access the 5620 SAM object.

Procedure A-4 An XML request fails

Check with your administrator to determine whether you have sufficient privileges to execute the request. If the request contains multiple requests, verify each request to isolate the problem.

Procedure A-5 Unable to perform an action using the 5620 SAM-O

Use the 5620 SAM GUI or the script manager to perform the action. See the *5620 SAM User Guide* and *5620 SAM Scripts and Templates Developer Guide* for more information.

Check with your administrator to determine whether you have sufficient privileges to perform the action. You receive the following exception if you are not assigned the appropriate scope of command for the 5620 SAM-O:

```
<?xml version="1.0" ?>
- <SOAP:Envelope
  xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
- <SOAP:Header>
- <header xmlns="xmlapi_1.0">
  <requestID>XmlApiClient:2</requestID>
  <requestTime>Apr 1, 2008 2:35:46 PM</requestTime>
  <responseTime>Apr 1, 2008 2:35:46 PM</responseTime>
</header>
</SOAP:Header>
- <SOAP:Fault>
```

```

<faultcode>SOAP:Client</faultcode>
<faultstring>[security] Users require OSS Management privileges to
use SAM-O</faultstring>
<faultactor>XmlApi</faultactor>
- <detail>
<requestID>XmlApiClient:2</requestID>
</detail>
</SOAP:Fault>
</SOAP:Envelope>

```

Procedure A-6 Identifying XML messages from specific users

Use the following procedure to enable the logging of the 5620 SAM-O XML request, response, and exception messages on the active or standby 5620 SAM servers. The entries in the XML message log files can help you identify the XML requests of a user, multiple users, or of all of the users. See [“Log file for XML requests, responses, and exceptions”](#) in chapter 9 for more information about the logging of XML messages.

You must update the nms-server.xml file on the 5620 SAM server with DEBUG logging to view OSS template information in 5620 SAM server logs.



Caution — The purpose of the XML message log is to debug activities in an OSS application development environment. The message log allows developers to evaluate the interaction of a third-party application with the 5620 SAM-O.

- 1 Log in to the main server station as the samadmin user.
- 2 Navigate to the server configuration directory, typically /opt/5620sam/server/nms/config.
- 3 Create a backup copy of the nms-server.xml file.
- 4 Open the nms-server.xml file using a plain-text editor.
- 5 Locate the <systemOssiLog> element.
- 6 Modify the attributes of the <systemOssiLog> element to enable the log option for an individual user or multiple users. The 5620 SAM creates a unique log file for each HTTP request and response associated with each 5620 SAM-O user. The request log contains the body of the SOAP message. The response log contains the entire SOAP envelope of the response.



Caution — Do not modify other nms-server.xml parameters. A modification may seriously affect 5620 SAM network management and performance.

- a To log the XML messages for all 5620 SAM-O users, enter:

```

<systemOssiLog
    allUsers="yes"
    path="../log/all_users"/>

```

The above example creates files named `ossiuserRequestn+.log` and `ossiuserResponsen+.log` in the `path/nms/log/individual/` directory

where

`user` is the 5620 SAM user name

`n+` is the incremental request number

`path` is the 5620 SAM server installation location, typically `/opt/5620sam/server`

- b** To log the XML messages for a single 5620 SAM-O user, enter:

```
<systemOssiLog  
  
    allUsers="no"  
  
    user="yes"  
  
    path=" ../log/individual"/>
```

The above example creates files named `ossiuserRequestn+.log` and `ossiuserResponsen+.log` in the `path/nms/log/individual/` directory

where

`user` is the 5620 SAM user name

`n+` is the incremental request number

`path` is the 5620 SAM server installation location, typically `/opt/5620sam/server`

- 7 Save and close the `nms-server.xml` file.
- 8 Open a console window.
- 9 Navigate to the server binary directory, typically `/opt/5620sam/server/nms/bin`.
- 10 Enter the following at the prompt:

```
bash$ ./nmsserver.bash read_config ↵
```

The main server reads the `nms-server.xml` file and puts the configuration changes into effect.

Procedure A-7 Receive a `java.lang.UnsupportedClassVersionError` when sending scripts using the PostXML tool

The PostXML tool requires Java version 1.4.2_02 or later. This problem occurs when you install Oracle after installing Java. Verify that the right Java version is in the path. The following error message is an example of the notification that you receive when you run the PostXML batch file:

```
Exception in thread "main" java.lang.UnsupportedClassVersionError:  
PostXML (Unsupported major.minor version 48.0)  
  
at java.lang.ClassLoader.defineClass0(Native Method)  
  
at java.lang.ClassLoader.defineClass(ClassLoader.java:495)
```



```
at
java.security.SecureClassLoader.defineClass(SecureClassLoader.java:1
10)

at java.net.URLClassLoader.defineClass(URLClassLoader.java:251)
at java.net.URLClassLoader.access$1(URLClassLoader.java:217)
at java.net.URLClassLoader$1.run(URLClassLoader.java:198)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(URLClassLoader.java:192)
at java.lang.ClassLoader.loadClass(ClassLoader.java:300)
at sun.misc.Launcher$AppClassLoader.loadClass(Launcher.java:290)
at java.lang.ClassLoader.loadClass(ClassLoader.java:256)
at java.lang.ClassLoader.loadClassInternal(ClassLoader.java:316)
```

Procedure A-8 Receive a java.net.ConnectException when sending scripts using the PostXML tool

You must verify that the correct 5620 SAM server is configured in the PostXML batch file. By default, the 5620 SAM server points to localhost. The following error message is an example of the notification that you receive when you run the PostXML batch file:

```
Exception in thread "main" java.net.ConnectException: Connection refused:
connect

at java.net.PlainSocketImpl.socketConnect(Native Method)
at java.net.PlainSocketImpl.doConnect(PlainSocketImpl.java:305)
at java.net.PlainSocketImpl.connectToAddress(PlainSocketImpl.java:171)
at java.net.PlainSocketImpl.connect(PlainSocketImpl.java:158)
at java.net.Socket.connect(Socket.java:452)
at java.net.Socket.connect(Socket.java:402)
at java.net.Socket.(Socket.java:309)
at java.net.Socket.(Socket.java:124)

at
org.apache.commons.httpclient.protocol.DefaultProtocolSocketFactory.createSo
cket(DefaultProtocolSocketFactory.java:118)

at
org.apache.commons.httpclient.HttpConnection.open(HttpConnection.java:683)

at
org.apache.commons.httpclient.HttpClient.executeMethod(HttpClient.java:662)

at
org.apache.commons.httpclient.HttpClient.executeMethod(HttpClient.java:529)
```

```
at PostXML.main(PostXML.java:136)
```

Procedure A-9 Receive a java.net.ConnectException when sending HTTP requests to the 5620 SAM-O server

You receive the following connection refused exception when the 5620 SAM-O server does not respond to HTTP requests:

```
java.net.ConnectException: Connection refused
```

The exception can be caused by the following server conditions:

- server is starting up and not ready to accept connections
- server session limit reached
- host name or IP address in the URL is incorrect
- maximum number of concurrent connections reached
- maximum number of active connections reached on user or user group

The maximum number of JMS connections is 10. The maximum number of concurrent HTTP OSS connections is 10 or more, depending on the server hardware. See the *5620 SAM Planning Guide* for information about platform sizing. If the number of HTTP OSS connections is exceeded, the client application receives an HTTP 503 message for maximum session reached, or an HTTP 903 message for maximum session per user.

Procedure A-10 The OSS client cannot connect with the HTTP or JMS server

Verify the following:

- The 5620 SAM-O client application points to the correct IP address and port of the servers.
- Firewalls between the 5620 SAM-O client and the servers are correctly configured.

- 1 Open the nms-server.xml file in the 5620 SAM install directory.
- 2 Verify that the IP addresses are correct.
- 3 Close the nms-server.xml file and save the changes.
- 4 Notify the 5620 SAM server of the changes made to the nms-server.xml file by typing:

```
nmsserver read_config
```

The changes to the `nms-server.xml` file are read by the 5620 SAM server after issuing the `read_config` command or restarting the server. See the *5620 SAM Planning Guide* for more information about firewall configuration.

- 5 If you still think that there are connectivity problems, perform the following:
- Use a standard ICMP ping to verify that you can ping the server from the OSS client station.
 - For XML requests, perform an XML ping using the PostXML utility. See section 5.1 for more information.
 - For JMS problems, try to create the same connection using JMS Test. See Procedure 4-3 for more information.
 - Verify that the required ports are open between the OSS client station and the server. See the *5620 SAM Planning Guide* for firewall configuration information.
-

Procedure A-11 The OSS client cannot perform find or findToFile requests

If you receive an exception indicating that the number of inventory connections has reached the maximum, the request was rejected. You can retry again later when the number of requests are reduced. This occurs when there are five find or findToFile concurrent requests on the 5620 SAM server.

B. 5620 SAM-O SDK library of samples

B.1 Description 5620 SAM-O SDK library of XML samples B-2

B.1 Description 5620 SAM-O SDK library of XML samples

The 5620 SAM-O SDK includes a library of XML scripts that provide configuration and network management samples to facilitate OSS integration with the 5620 SAM-O. The SDK consists of a Basic SDK and an Advanced SDK. These samples can be used as a base to build 5620 SAM-O requests.



Warning — Developers should thoroughly test their XML requests before network deployment.

The Basic SDK contains XML scripts that provide examples of how to perform general procedures, such as configure, create, delete, find and filter on 5620 SAM objects. The Advanced SDK provides more comprehensive examples for the entire network management domain that the 5620 SAM supports, such as configuration, fault, inventory, OAM, statistics, and proprietary 5620 SAM methods.



Note — Contact your Alcatel-Lucent OIPS representative for more information about how to obtain the 5620 SAM-O SDK library of sample XML scripts. If OSS developers need assistance developing XML scripts that are not included in the SDK, send a request should be sent to developer.support@alcatel-lucent.com.

Basic SDK

The Basic SDK contains the following XML scripts:

- ConfigureObject.xml
- CreateObject.xml
- DeleteObject.xml
- FilterExample.xml
- Find.xml
- FindToFile.xml
- Ping.xml
- CreateObjectWithResultFilter.xml
- FilterExamples-FilteringBasedOnChildren.xml
- ResultFilterExample-ExcludingChildren.xml
- ResultFilterExample-FilterChildren.xml
- ResultFilterExample-RecursiveResultFilter.xml

Advanced SDK

The Advanced SDK contains a comprehensive set of XML scripts that can be categorized into the following:

- Configuration
- Fault
- Inventory
- Miscellaneous
- OAM
- Server
- Statistics

Configuration

Samples that allow you to perform configuration on the following objects in the 5620 SAM:

- CPAM
- discovery
- equipment
- GNE
- interfaces
- policies
- protocols
- router
- script management
- services
- service tunnels
- shared queuing
- SRLG
- static routes
- subscribers

Fault

Samples that allow you to perform fault management operations, such as ackFaults, FindFaults, and clearFaults.

Inventory

Samples that allow you to perform find or findToFile operations on objects in the following areas in the 5620 SAM:

- equipment
- policies
- services
- service tunnels
- static routes
- subscribers

Miscellaneous

Samples for specific methods in the following areas in the 5620 SAM:

- CLI
- Deployer

OAM

Samples that allow you to perform OAM tests, such as LspPing and MacPing.

Server

Samples that allow you to perform general server operations on the 5620 SAM server, such as accessing the 5620 SAM release and pinging the server.

Statistics

Samples that allow you to perform statistic operations to collect statistics; for example, registerLogToFile and FindToFile.

Customer documentation and product support



Customer documentation

<http://www.alcatel-lucent.com/myaccess>

Product manuals and documentation updates are available at [alcatel-lucent.com](http://www.alcatel-lucent.com). If you are a new user and require access to this service, please contact your Alcatel-Lucent sales representative.



Technical Support

<http://support.alcatel-lucent.com>



Documentation feedback

documentation.feedback@alcatel-lucent.com



© 2011-2012 Alcatel-Lucent. All rights reserved.

3HE 06498 AAAG TQZZA Edition 01