Alcatel-Lucent 5620

SERVICE AWARE MANAGER | RELEASE 9.0 R2

3GPP OSS INTERFACE DEVELOPER GUIDE

3HE 06509 AAAB TQZZA Edition 01

Alcatel-Lucent Proprietary
This document contains proprietary information of Alcatel-Lucent and is not to be disclosed or used except in accordance with applicable agreements.
Copyright 2011 © Alcatel-Lucent. All rights reserved.

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 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.

When printed by Alcatel-Lucent, this document is printed on recycled paper.

Alcatel-Lucent License Agreement

SAMPLE END USER LICENSE AGREEMENT

1. LICENSE

1.1Subject 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.

1.2Customer 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

2.1Customer 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.

2.2All 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

3.1This 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.

8.4The 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 FREEWARE OR SHAREWARE.

8.5Alcatel-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.

8.6All 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.

8.7If 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.

8.8Customer 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.

Note — You can use the Search function of Acrobat Reader (File→Search) to find a term in a PDF of this 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. For more information, see the Help for Acrobat Reader.

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 documentation suite.

Table 1 5620 SAM customer documentation suite

Guide	Description
5620 SAM core documentation	
5620 SAM Planning Guide	The *5620 SAM Planning Guide* provides information about 5620 SAM scalability and recommended hardware configurations.

(1 of 4)

Guide	Description
5620 SAM | 5650 CPAM Installation and Upgrade Guide	The *5620 SAM | 5650 CPAM Installation and Upgrade Guide* provides OS considerations, configuration information, and procedures for the following: • 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
5620 SAM User Guide	The *5620 SAM User Guide* 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 *5620 SAM User Guide* uses a task-based format. Each chapter contains: • a workflow that describes the steps for configuring and using the functionality • detailed procedures that list the configurable parameters on the associated forms 5620 SAM management information specific to LTE network elements is covered in the *5620 SAM LTE User Guide* and *5620 SAM LTE RAN User Guide*. 5620 SAM management information specific to 1830 PSS network elements is covered in the *5620 SAM Optical User Guide*.
5620 SAM Parameter Guide	The *5620 SAM Parameter Guide* provides: • 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 *5620 SAM User Guide* and the parameter descriptions in the *5620 SAM Parameter Guide*. See Procedure 2 for more information. Parameters specific to LTE network elements are covered in the *5620 SAM LTE Parameter Reference*. Parameters specific to 1830 PSS network elements are covered in the *5620 SAM Optical Parameter Reference*.
5620 SAM Statistics Management Guide	The *5620 SAM Statistics Management Guide* provides information about how to configure performance and accounting statistics collection and how to view counters using the 5620 SAM. Network examples are included.
5620 SAM Scripts and Templates Developer Guide	The *5620 SAM Scripts and Templates Developer Guide* 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: • CLI scripting, XML, and the Velocity engine • basic scripting or programming • 5620 SAM functions

5620 SAM Troubleshooting Guide	The *5620 SAM Troubleshooting Guide* provides task-based procedures and user documentation to:
	• help resolve issues in the managed and management networks
	• identify the root cause and plan corrective action for:
	  • 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:
	  • network management LANs
	  • network management platforms and operating systems
	  • 5620 SAM client GUIs and client OSS applications
	  • 5620 SAM servers
	  • 5620 SAM databases

(2 of 4)

Guide	Description
5620 SAM Maintenance Guide	The *5620 SAM Maintenance Guide* provides procedures for: • generating baseline information for 5620 SAM applications • performing daily, weekly, monthly, and as-required maintenance activities for 5620 SAM-managed networks
5620 SAM Integration Guide	The *5620 SAM Integration Guide* provides procedures to allow the 5620 SAM to integrate with additional components.
5620 SAM System Architecture Guide	The *5620 SAM System Architecture Guide* 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.
5620 SAM Supervision Module User Guide	The *5620 SAM Supervision Module User Guide* 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.
5620 SAM Network Element Compatibility Guide	The *5620 SAM Network Element Compatibility Guide* provides release-specific information about the compatibility of managed device features in 5620 SAM releases.
5620 SAM Release Description	The *5620 SAM Release Description* provides information about the new features associated with a 5620 SAM software release.
5620 SAM Glossary	The *5620 SAM Glossary* defines terms and acronyms used in all of the 5620 SAM documentation, including 5620 SAM LTE documentation.
5620 SAM XML OSS Interface Developer Guide	The *5620 SAM XML OSS Interface Developer Guide* provides information that allows you to: • 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
5620 SAM LTE documentation	
5620 SAM LTE User Guide	The *5620 SAM LTE User Guide* 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 *5620 SAM LTE User Guide* before you attempt to use the 5620 SAM in your LTE network.
5620 SAM LTE RAN User Guide	The *5620 SAM LTE RAN User Guide* describes how to discover, configure, and manage the 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 *5620 SAM LTE RAN User Guide* before you attempt to use the 5620 SAM in your LTE network.
5620 SAM LTE Parameter Reference	The *5620 SAM LTE Parameter Reference* provides a list of all LTE ePC and LTE RAN parameters supported in the 5620 SAM.
5620 SAM LTE Alarm Reference	The *5620 SAM LTE Alarm Reference* provides a list of LTE ePC and LTE RAN alarms that can be reported in the 5620 SAM GUI.
5620 SAM 3GPP OSS Interface Developer Guide	The *5620 SAM 3GPP OSS Interface Developer Guide* 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.
5620 SAM 3GPP OSS Interface Compliance Statements	The *5620 SAM 3GPP OSS Interface Compliance Statements* document describes the compliance of the 5620 SAM 3GPP OSS interface with the 3GPP standard.
5620 SAM LTE RAN Release Description	The *5620 SAM LTE RAN Release Description* provides information about the LTE RAN features associated with the release.

(3 of 4)

Guide	Description
5620 SAM optical documentation	
5620 SAM Optical User Guide	The *5620 SAM Optical User Guide* 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 *5620 SAM Optical User Guide* before you attempt to use the 5620 SAM in your network.
5620 SAM Optical Parameter Reference	The *5620 SAM Optical Parameter Reference* provides a list of all optical device parameters supported in the 5620 SAM.
5620 SAM Optical Alarm Reference	The *5620 SAM Optical Alarm Reference* provides a list of optical device alarms that can be reported in the 5620 SAM GUI.

(4 of 4)

Procedure 1 To find the 5620 SAM user documentation

The user documentation is available from the following sources:

- the User_Documentation directory on the product DVD-ROM
- Help→5620 SAM User Documentation in the 5620 SAM client GUI main menu

Note — Users of Mozilla browsers may receive an error message when using the User Documentation Index page (index.html) to open 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 offline storage (Mozilla Firefox) or cache (Mozilla 1.7) values to 100 Mbytes to eliminate the error message.

Procedure 2 To view parameter descriptions from the *5620 SAM User Guide*

You can click on a parameter name in a *5620 SAM User Guide* procedure to open the matching parameter description in the *5620 SAM Parameter Guide*. Ensure the following conditions are true beforehand:

- the *5620 SAM Parameter Guide* and *5620 SAM User Guide* are located in the same directory
- Adobe Reader Release 5.0 or later is installed

1. To view a parameter description when both the *5620 SAM User Guide* and the *5620 SAM Parameter Guide* are open in Adobe Acrobat, click on the parameter name in the *5620 SAM User Guide*.

   The parameter description is displayed in the *5620 SAM Parameter Guide*.

2. To view a parameter description when only the *5620 SAM User Guide* is open in Adobe Acrobat:

   i. Click on a parameter name in a procedure in the *5620 SAM User Guide*. The *5620 SAM User Guide* closes and the *5620 SAM Parameter Guide* opens to display the parameter description.

   ii. Double-click on the Previous View button in Adobe Acrobat (or press Alt + ←) to re-open the *5620 SAM User Guide*. The *5620 SAM User Guide* opens and displays the parameter from step i.

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

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	*hostname*

(1 of 2)

Convention	Description	Example
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

(2 of 2)

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**

5620 SAM documentation suite ix
Procedure 1 To find the 5620 SAM user documentation xii
Procedure 2 To view parameter descriptions from the 5620 SAM User Guide xiii
Prerequisites xiii
Conventions xiii
Procedures with options or substeps xiv
Measurement conventions xiv
Important information xv

5620 SAM 3GPP OSS interface

1 — Introduction **1-1**

1.1 5620 SAM 3GPP OSS interface overview 1-2
1.2 CORBA overview 1-2
1.3 3GPP overview 1-2

2 — IRPs **2-1**

2.1 IRP overview 2-2
2.2 IRP specification layers 2-2
2.3 IRP components 2-2
2.4 5620 SAM 3GPP OSS interface IRPS 2-2
Entry Point IRP 2-4
Alarm IRP 2-6

Basic CM IRP 2-12
Notification IRP 2-14
Communication Surveillance IRP 2-18
Kernel IRP 2-19
Generic IRP 2-20

3 — Communicating with OSS clients 3-1

3.1 Initial access 3-2
3.2 Event monitoring 3-2
Notification IRP initialization 3-2
Subscribing to event notifications 3-3
Procedure 3-1 To subscribe to and unsubscribe from event notifications 3-3
Filtering 3-4
Connection monitoring 3-5
Error recovery 3-5
3.3 Security 3-5
User security 3-6
SSL security 3-6
Firewall support 3-6
Procedure 3-2 To configure a 3GPP OSS interface to operate through a firewall 3-7
3.4 Redundancy 3-8
3.5 Multiple 3GPP release support 3-8

4 — 5620 SAM 3GPP OSS domains 4-1

4.1 Fault management 4-2
4.2 Workflow to obtain an alarm list 4-2
4.3 Workflow to acknowledge an alarm list 4-2
4.4 Inventory management 4-2
Basic inventory 4-3
3GPP compliant Inventory File generation and description 4-3
Automatic generation of Inventory File 4-3
3GPP Inventory File properties description 4-4
Inventory file attributes 4-4
Sample 3GPP Inventory File 4-5
4.5 Workflow to obtain a list of NEs 4-6

Appendices

A. 5620 SAM-O 3GPP code samples A-1

A.1 Code samples A-2
Fault management code samples A-2
Basic inventory management code samples A-2

B. Standards compliance B-1

B.1 Standards compliance B-2

C. Troubleshooting C-1

C.1 3GPP OSS troubleshooting overview C-2
C.2 Diagnosing typical 3GPP installation problems on the 5620 SAM server C-2
Typical installation problem..... C-2
3GPP OSS user and password configuration problems C-3
Procedure C-1 To update the cnbi.properties file in the 5620 SAM C-3
Troubleshooting 3GPP OSS SSL communication problems C-3
Procedure C-2 To verify that the 3GPP OSS Interface is successfully connected to the 5620 SAM server C-4
Verifying that messages are being received by the 3GPP OSS interface from the 5620 SAM C-4
Firewall preventing OSS client access to the 3GPP OSS Interface C-5
Problems referencing the Entry Point IOR C-5
Incorrect configuration of the System DN C-5
Problems receiving alarms and events when SPAN is configured C-5
C.3 Useful logs..... C-5

5620 SAM 3GPP OSS interface

1 — Introduction

2 — IRPs

3 — Communicating with OSS clients

4 — 5620 SAM 3GPP OSS domains

1 — Introduction

1.1 5620 SAM 3GPP OSS interface overview *1-2*

1.2 CORBA overview *1-2*

1.3 3GPP overview *1-2*

1.1 5620 SAM 3GPP OSS interface overview

The 3GPP OSS Interface is the CORBA-based API that provides a 3GPP-compliant layer for communication with a 5620 SAM-managed LTE network using a client application. The interface includes a 3GPP CORBA client adapter. The CORBA solution set includes the JacORB ORB implementation.

The supported 3GPP OSS interface network management functions include the following:

- alarm and event monitoring
- inventory management

The following sections provide brief introductions to CORBA and 3GPP.

1.2 CORBA overview

Common Object Request Broker Architecture, or CORBA, is a standard architecture for distributed object systems. It allows a distributed, heterogeneous collection of objects to interoperate. Its framework provides access across platforms, operating systems, and programming languages. It is an object-oriented architecture that has the characteristics of object-oriented systems. It was developed by the Object Management Group, or OMG.

The following components of the CORBA specification enable interoperation with other systems:

- Interface Definition Language, or IDL
- Object Request Broker, or ORB
- GIOP and IIOP protocols

The IDL provides a language-neutral way of specifying the classes, called interfaces in the IDL, of the objects within the application. An interface consists of a set of named operations and the operation parameters.

The ORB is the core of the CORBA architecture. It establishes client-server relationships and provides platform independence to distributed CORBA objects. The ORB acts as the mediator and translator that enables client applications to request the services of CORBA objects and servers to make object services available to client applications.

The General Inter-ORB Protocol, or GIOP, defines the message formats for object communication in a distributed environment. The Internet Inter-ORB Protocol, or IIOP, is an implementation of GIOP that operates over TCP/IP.

1.3 3GPP overview

The 3rd Generation Partnership Project, or 3GPP, is a collaboration between groups of telecommunications associations to make a globally applicable third-generation mobile phone system specification within the scope of the International Mobile Telecommunications-2000 project of the International Telecommunication Union, or ITU.

The term “3GPP specification” includes all GSM specifications, for example GPRS, EDGE, W-CDMA, and LTE, including LTE-Advanced. The following terms are also used to describe networks that use the 3GPP specifications:

- UTRAN and UMTS in Europe
- FOMA in Japan

For more information about 3GPP, see the 3GPP website at www.3gpp.org.

2 — IRPs

2.1 IRP overview *2-2*

2.2 IRP specification layers *2-2*

2.3 IRP components *2-2*

2.4 5620 SAM 3GPP OSS interface IRPS *2-2*

2.1 IRP overview

The 3GPP management interface concept called Integration Reference Point, or IRP, promotes the adoption of standardized management interfaces in telecommunication networks. The IRP concept and associated methodology use protocol- and technology-neutral modeling methods, and protocol-specific solution sets.

The 3GPP publishes IRP specifications for operations and notifications for telecommunication domains such as alarm, configuration, and performance management. Detailed IRP specifications are available on the 3GPP website at www.3gpp.org/specifications.

2.2 IRP specification layers

A 3GPP IRP definition consists of layers that include the following:

- IRP requirements—high-level IRP description
- information service—detailed IRP description
- solution sets—low-level descriptions of the APIs for an IRP

The solution sets describe the low-level interoperability mechanisms of the API that the element managers and network managers must conform to. For the 5620 SAM 3GPP OSS interface, the supported solution set uses CORBA that is compliant with the R7 and R8 3GPP standards. See Appendix B for more information.

2.3 IRP components

The IRP specification describes the following main components:

- IRP agent—encapsulates a well-defined subset of network or NE functions; it interacts with IRP managers using one or more IRPs
- IRP manager—models a user of IRP agents and interacts with IRP agents through IRPS

The 5620 SAM is an IRP agent, and an OSS application is the IRP manager

2.4 5620 SAM 3GPP OSS interface IRPS

The 3GPP has defined a number of IRPs that map to telecommunications management domains in a 3GPP network. The 5620 SAM 3GPP OSS interface supports a subset of these IRPs that includes the following:

- Entry Point IRP—allows OSS applications to discover the management system and IRPs
- Notification IRP—allows OSS applications to subscribe to notifications and manage subscriptions
- Alarm IRP—allows OSS applications to detect and manage alarms and faults
- Communication Surveillance IRP—provides a communication heartbeat for monitoring OSS application communication with the network manager

- Basic CM IRP—allows OSS applications to control and monitor NE configurations and network resources
- Kernel IRP—provides OSS applications with a list of the version of all IRPs in the network and also defines notification on NEs for object creation, deletion, attribute value changes, state changes, and synchronization information
- Generic IRP—provides OSS applications with IRP version, method and notification profile information for a specific IRP

Note — Although the 3GPP OSS interface supports the IRPs described above, it may not support all interfaces and methods defined in the 3GPP standard specifications. See Appendix B for more information.

Because the 3GPP OSS interface uses the CORBA IRP solution set, the IRPs are defined by CORBA IDLs. These IDLs are located in the following directory on a 5620 SAM main server:

```
installation_directory/nms/integration/SAM_O/3GPP
```

where *installation_directory* is the 5620 SAM main server installation directory, typically /opt/5620sam/server

Each IRP supported by an IRP agent supports a set of methods and notifications. The 3GPP OSS interface IDLs include interfaces and methods that have operation and notification definitions. The 5620 SAM 3GPP OSS interface compliance statement includes information about whether an interface method is mandatory or optional, and 3GPP standard versions with which it is compliant.

Note — The Basic CM and Generic IRPs do not have notification interface definitions.

For example, Table 2-1 lists the interfaces that are defined for the Communication Surveillance, or CS, IRP.

Table 2-1 CS IRP compliance with 3GPP versions

Interface name	Method name	Support Qualifier	3GPP R7	3GPP R8
CSIRPOperations_1	getHeartbeatPeriod	Mandatory	Y	Y
	triggerHeartbeat	Mandatory	N	N
CSIRPOperations_2	setHeartbeatPeriod	Optional	N	N
CSIRPNotifications	notifyHeartbeat	Mandatory	Y	Y

Entry Point IRP

It is difficult for a network manager to discover all IRPs in an environment that contains several managed systems, or if there are multiple IRPs associated with each managed system. The Entry Point IRP provides a mechanism for the network manager to discover the managed systems and associated IRPs.

The Entry Point IRP allows a network manager to do the following:

- Discover IRP information that includes the DN, supported IRP versions, and management scope, the set of network resources in the managed system to which it has access.
- Obtain the references of the IRPs for a specific IRP version using the DN of the IRP that supports the required IRP version. The network manager can use the Entry Point IRP to tell the managed system which references it does not use. The managed system may release the references, if it is designed to do so.

During initialization, the 3GPP OSS interface generates an IOR for the Entry Point IRP and stores it in the following file on a 5620 SAM main server:

```
$CNBI_HOME/ior/EPIRP.ior
```

Code 2-1 shows an example of IRP reference retrieval using the Entry Point IRP.

Code 2-1: Entry Point IRP reference retrieval example

```
String iorDir = “$CNBI_HOME/ior directory”;
String iorFile = iorDir + File.separator + "EPIRP.ior";

FileInputStream fis = new FileInputStream(iorFile);
BufferedReader reader = new BufferedReader(new
InputStreamReader(fis));
String ior = reader.readLine();
reader.close();

try
{
org.omg.CORBA.Object entryPointObj = orb.string_to_object(ior);
entryPointIRP = EPIRPHelper.narrow(entryPointObj);
}
catch (Exception e)
{
e.printStackTrace();
}
```

The Entry Point IRP includes the following methods:

- getIRPOutline—used to obtain information about all supported IRPs
- getIRPReference—used to request a CORBA reference for a specific IRP

getIRPOutline method

Code 2-2 shows the getIRPOutline method definition.

Code 2-2: getIRPOutline method

```
EPIRPConstDefs::Result get_irp_outline(
```

```
      in ManagedGenericIRPConstDefs::VersionNumber irp_version,
      out EPIRPConstDefs::SupportedIRPList supported_irp_list
   )
   raises (GetIRPOutline, InvalidIRPVersion);
```

Note — The interface returns IRP information for only the local IRP agent.

Code 2-3 shows an example of getIRPOutline method invocation.

Code 2-3: getIRPOutline method invocation example

```
try
{
String irp_version = "";
SupportedIRPListHolder supported_irp_list = new
SupportedIRPListHolder();
Result result =
entryPointIRP.get_irp_outline(irp_version,supported_irp_list);
SupportedIRP supportedIRP = supported_irp_list.value[0];
}
catch (Exception e)
{
e.printStackTrace();
}
```

getIRPReference method

Code 2-4 shows the getIRPReference method definition.

Code 2-4: getIRPReference method

```
   EPIRPConstDefs::Result get_irp_reference(
      in EPIRPConstDefs::ManagerIdentifier manager_identifier,
      in EPIRPConstDefs::DN system_dn,
      in EPIRPConstDefs::IRPId irp_id,
      out string irp_reference
   )
   raises (GetIRPReference,
           ManagedGenericIRPSystem::InvalidParameter);
```

Code 2-5 shows an example of getIRPReference method invocation for retrieving the reference to the Alarm IRP.

Code 2-5: getIRPReference method invocation example

```
try
{
entryPointIRP.get_irp_reference("n.a.", system_dn, "32.111-3 V8.0",
irp_reference);
org.omg.CORBA.Object alarmObj =
orb.string_to_object(irp_reference.value);
alarmIRP = AlarmIRPHelper.narrow(alarmObj);
}
catch (Exception e)
```

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

Note — The systemDn is a configurable value in the cnbi.properties file.

Alarm IRP

An evaluation of NE and network health requires the detection of faults in the network and the subsequent forwarding of associated alarms to the element manager or network manager. Depending on the nature of a fault, there may be a change in the operational state of the affected logical or physical resources.

Alarm and state-change detection and notification are essential. A list of the current network alarms, operational state information, and the associated history data are required for further fault analysis. Additionally, test procedures can be used to obtain more detailed information, if required, and to verify an alarm or operational state to ensure correct NE operation and the proper use of resources.

Contact Alcatel-Lucent technical support for a list of the notifications that the alarm IRP can raise.

Code 2-6 shows an example of Alarm IRP reference retrieval using the Entry Point IRP.

Code 2-6: Alarm IRP reference retrieval example

```
try
{
entryPointIRP.get_irp_reference("n.a.", system_dn, "32.111-3 V8.0",
irp_reference);
org.omg.CORBA.Object alarmObj =
orb.string_to_object(irp_reference.value);
alarmIRP = AlarmIRPHelper.narrow(alarmObj);
}
catch (Exception e)
{
e.printStackTrace();
}
```

The Alarm IRP includes the following methods:

- acknowledgeAlarm—acknowledges one or more alarms
- clearAlarms—clears one or more alarm instances in an alarm list
- commentAlarms—records a comment in one or more alarm information instances in an alarm list
- getAlarmList—returns the set of current alarms
- getAlarmCount—retrieves a list of alarm counts, by severity, for alarms in raised, cleared and acknowledgement states

acknowledgeAlarms method

Code 2-7 shows the acknowledgeAlarms method definition.

Code 2-7: acknowledgeAlarms method

```
ManagedGenericIRPConstDefs::Signal acknowledge_alarms (
         in AlarmIRPConstDefs::AlarmInformationIdAndSevSeq
            alarm_information_id_and_sev_list,
         in string ack_user_id,
         in ManagedGenericIRPConstDefs::StringOpt ack_system_id,
         out AlarmIRPConstDefs::BadAcknowledgeAlarmInfoSeq
            bad_ack_alarm_info_list
      )
      raises (AcknowledgeAlarms,
ManagedGenericIRPSystem::ParameterNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);
```

The method updates the specified alarm in the following ways:

- sets the Acknowledgement State to Acknowledged
- adds the ackUserId and ackSystemId values as comments to the alarm
- sets the Acknowledgement Time of the alarm to the current time

Note 1 — The method performs no verification of the provided ackUserId and ackSystemId values.

Note 2 — A perceivedSeverity value in the request is ignored; an alarm is acknowledged based on the alarm ID only.

Code 2-8 shows an example of acknowledge_alarms method invocation.

Code 2-8: acknowledge_alarms method invocation example

```
BadAcknowledgeAlarmInfoSeqHolder badackalarmholder = new
BadAcknowledgeAlarmInfoSeqHolder();
StringOpt opt = new StringOpt();
opt.value("system_id");
AlarmInformationIdAndSev alarm = new AlarmInformationIdAndSev();
AlarmInformationIdAndSev alarm2 = new AlarmInformationIdAndSev();

ShortOpt op = new ShortOpt();
op.value((short) 1);
alarm.alarm_id = "1";
alarm.perceived_severity = op;
alarm2.alarm_id = "2";
alarm2.perceived_severity = op;
try
{
alarmIRP.acknowledge_alarms(new AlarmInformationIdAndSev[] { alarm,
alarm2 }, "user_id", opt, badackalarmholder);
BadAcknowledgeAlarmInfo[] info = badackalarmholder.value;
}
catch (Exception e)
{
Log.error(this, "ERROR", e);
```

```
fail(e.getMessage());
}
```

If an acknowledgeAlarms operation fails, the list of errors, which includes the alarm identifiers and error codes, is stored in the bad alarm information object. This object is defined as a CORBA holder, and has to be created and passed by reference to the method.

clearAlarms method

Code 2-9 shows the clearAlarms method definition.

Code 2-9: clearAlarms method

```
      ManagedGenericIRPConstDefs::Signal clear_alarms (
         in AlarmIRPConstDefs::AlarmInformationIdSeq
alarm_information_id_list,
         in string clear_user_id,
         in ManagedGenericIRPConstDefs::StringOpt clear_system_id,
         out AlarmIRPConstDefs::BadAlarmInformationIdSeq
            bad_alarm_information_id_list
      )
      raises (ClearAlarms,
ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::ParameterNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);
```

Note — The method performs no verification of the provided clearUserId and clearSystemId values.

Code 2-10 shows an example of clear_alarms method invocation.

Code 2-10: clear_alarms method invocation example

```
BadAlarmInformationIdSeqHolder badclearalarmholder = new
BadAlarmInformationIdSeqHolder();
StringOpt opt = new StringOpt();
opt.value("system_id");
ShortOpt op = new ShortOpt();
op.value((short) 1);
try
{
alarmIRP.clear_alarms(new String[]{"1", "2"}, "user_id", opt,
badclearalarmholder);
BadAlarmInformationId[] info = badclearalarmholder.value;
}
catch (Exception e)
{
e.printStackTrace();
fail();
}
```

If a clear_alarms operation fails, the list of errors, which includes the alarm identifiers and error codes, is stored in the bad alarm information object. This object is defined as a CORBA holder, and has to be created and passed by reference to the method.

commentAlarms method

Code 2-11 shows the commentAlarms method definition.

Code 2-11: commentAlarms method

```
      ManagedGenericIRPConstDefs::Signal comment_alarms (
         in AlarmIRPConstDefs::AlarmInformationIdSeq
alarm_information_id_list,
         in string comment_user_id,
         in ManagedGenericIRPConstDefs::StringOpt comment_system_id,
         in string comment_text,
         out AlarmIRPConstDefs::BadAlarmInformationIdSeq
            bad_alarm_information_id_list
      )
      raises (CommentAlarms,
ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::ParameterNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);
```

Note — The method performs no verification of the provided commentUserId and commentSystemId values.

Code 2-12 shows an example of comment_alarms method invocation.

Code 2-12: comment_alarms method invocation example

```
BadAlarmInformationIdSeqHolder badcommentalarmholder = new
BadAlarmInformationIdSeqHolder();
StringOpt opt = new StringOpt();
opt.value("system_id");
ShortOpt op = new ShortOpt();
op.value((short) 1);
try
{
alarmIRP.comment_alarms(new String[]{"1", "2"}, "user_id", opt,
"alarm comment", badcommentalarmholder);
BadAlarmInformationId[] info = badcommentalarmholder.value;

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

If a commentAlarms operation fails, the list of errors, which includes the alarm identifiers and error codes, is stored in the bad alarm information object. This object is defined as a CORBA holder, and has to be created and passed by reference to the method.

getAlarmList method

Code 2-13 shows the getAlarmList method definition.

Code 2-13: getAlarmList method

```
      AlarmIRPConstDefs::AlarmInformationSeq get_alarm_list (
         in ManagedGenericIRPConstDefs::StringOpt filter,
         in AlarmIRPConstDefs::DNOpt base_object,
         out boolean flag,
         out AlarmInformationIterator iter
      )
      raises (GetAlarmList, FilterComplexityLimit,
 ManagedGenericIRPSystem::ParameterNotSupported,
             ManagedGenericIRPSystem::InvalidParameter);
```

Note — This operation is only supported in synchronous mode, which means that the data is returned using an iterator and not using a notification.

The returned alarm list contains all alarms in the current alarm list. Depending on the network manager fault management configuration, this may include the following:

- alarms that have a perceivedSeverity other than Cleared, and alarms that have a perceivedSeverity of Cleared but are not acknowledged
- only alarms that have a perceivedSeverity other than Cleared

The alarm life cycle reported by the interface is provided by the network manager.

Code 2-14 shows an example of get_alarm_list method invocation.

Code 2-14: get_alarm_list method invocation example

```
StringOpt filter = new StringOpt();
filter.value("$.filterable_data(h) == 1");
DNOpt base_object = new DNOpt();
base_object.value(base);
BooleanHolder flag = new BooleanHolder();
AlarmInformationIteratorHolder iter = new
AlarmInformationIteratorHolder();
alarmIRP.get_alarm_list(filter, base_object, flag, iter);
AlarmInformationIterator iterator = iter.value;
List<StructuredEvent> resultList = new ArrayList<StructuredEvent>();
EventBatchHolder alarm_informations = new EventBatchHolder();
while (iterator.next_alarm_informations((short) 10,
alarm_informations))
{
resultList.addAll(Arrays.asList(alarm_informations.value));
}
resultList.addAll(Arrays.asList(alarm_informations.value));
```

An iterator object is created and passed as a parameter to the get_alarm_list method. Alarm batches are retrieved using iterator calls to the next_alarm_informations method, which returns a Boolean value; if the value is true, batches are pending.

When the 5620 SAM main server becomes available after being unavailable for a time, the 3GPP OSS interface re-establishes a connection to the server. If the interface detects missed events, it sends a notifyAlarmListRebuilt notification to the OSS clients to inform them that the alarm list is again reliable. See chapter 3 for information about connection monitoring and redundancy.

getAlarmCount method

Code 2-15 shows the getAlarmCount method definition.

Code 2-15: getAlarmCount method

```
      void get_alarm_count (
         in ManagedGenericIRPConstDefs::StringOpt filter,
         out unsigned long critical_count,
         out unsigned long major_count,
         out unsigned long minor_count,
         out unsigned long warning_count,
         out unsigned long indeterminate_count,
         out unsigned long cleared_count
      )
  raises (GetAlarmCount, FilterComplexityLimit,
          ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::ParameterNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);
```

Code 2-16 shows an example of get_alarm_count method invocation.

Code 2-16: get_alarm_count method invocation example

```
StringOpt filter = new StringOpt();
IntHolder critical_count = new IntHolder();
IntHolder major_count = new IntHolder();
IntHolder minor_count = new IntHolder();
IntHolder warning_count = new IntHolder();
IntHolder indeterminate_count = new IntHolder();
IntHolder cleared_count = new IntHolder();
try
{
filter.value("");
alarmIRP.get_alarm_count(filter, critical_count, major_count,
minor_count, warning_count, indeterminate_count, cleared_count);
}
catch (Exception e)
{
e.printStackTrace();
}
```

The counters are defined as CORBA holders, and are created and passed by reference. The counter values are retrievable after the method invocation.

Basic CM IRP

Configuration Management, or CM, provides the ability to ensure correct and effective network operation. Basic CM actions control and monitor the NE and network resource configurations, and can be initiated by an operator or by functions in an NE operations system.

The Basic CM IRP allows OSS clients to invoke methods that retrieve basic data about the 3GPP NEs in the network. This data can be associated with data from other functions, for example, alarm management.

The Basic CM IRP supports the following NE attributes:

- Managed Element ID
- User Label
- Type
- Managed By
- User Defined State
- Software Version
- Location Name
- Vendor Name

Supported NE Types

The 5620 SAM 3GPP OSS interface supports the retrieval of data for the following NE types:

- eNodeB
- 9471 MME

Code 2-17 shows an example of Basic CM IRP reference retrieval using the Entry Point IRP.

Code 2-17: Basic CM IRP reference retrieva example

```
try
{
entryPointIRP.get_irp_reference(manager_identifier, system_dn,
BasicCMIRPImpl.NAME_TO_BIND, irp_reference);
org.omg.CORBA.Object basicCMObj =
orb.string_to_object(irp_reference.value);
basicCMIRP = _BasicCmIrpOperationsHelper.narrow(basicCMObj);
}
catch (Exception e)
{
e.printStackTrace();
}
```

The Basic CM IRP includes the following methods:

- getMOAttributes
- getContainment

The getMOAttributes and getContainment methods are included in the same CORBA solution set definition.

getMOAttributes and getContainment methods

Code 2-18 shows the getMOAttributes and getContainment methods definition.

Code 2-18: getMOAttributes and getContainment methods

```
BasicCmInformationIterator find_managed_objects(
         in BasicCMIRPConstDefs::DN base_object,
         in BasicCMIRPConstDefs::SearchControl search_control,
         in BasicCMIRPConstDefs::AttributeNameSet
requested_attributes)
      raises (
         FindManagedObjects,
         ManagedGenericIRPSystem::ParameterNotSupported,
         ManagedGenericIRPSystem::InvalidParameter,
         ManagedGenericIRPSystem::ValueNotSupported,
         ManagedGenericIRPSystem::OperationNotSupported,
         UndefinedMOException,
         IllegalDNFormatException,
         UndefinedScopeException,
         IllegalScopeTypeException,
         IllegalScopeLevelException,
         IllegalFilterFormatException,
           FilterComplexityLimit);
```

Code 2-19 shows an example of getMOAttributes and getContainment methods invocation.

Code 2-19: getMOAttributes and getContainment methods invocation example

```
String base_object = "";
BasicCmInformationIterator cmIterator = null;
ResultSetHolder batchHolder = new ResultSetHolder();
ResultContents result_contents =
ResultContents.NAMES_AND_ATTRIBUTES;
// remaining SearchControl fields are not supported
SearchControl search_control = new SearchControl(ScopeType.BASE_ONLY,
0, "useless filter", result_contents);
try
{
// empty base_object, NAMES_AND_ATTRS
cmIterator = basicCMIRP.find_managed_objects(base_object,
search_control, new String[]{});
boolean isDataPending =
cmIterator.next_basic_cm_informations((short)10, batchHolder);
Result[] result = batchHolder.value;

// given base_object, NAMES_AND_ATTRS
base_object =
"SubNetwork=IPSubNetwork,ManagedElement=network:135.159.1.1";
cmIterator = basicCMIRP.find_managed_objects(base_object,
search_control, new String[]{});
isDataPending = cmIterator.next_basic_cm_informations((short)1,
batchHolder);
result = batchHolder.value;

// empty base_object, NAMES only
base_object = "";
result_contents = ResultContents.NAMES;
search_control = new SearchControl(ScopeType.BASE_ONLY, 0, "useless
filter", result_contents);
cmIterator = basicCMIRP.find_managed_objects(base_object,
search_control, new String[]{});
```

```
isDataPending = cmIterator.next_basic_cm_informations((short)10,
batchHolder);
result = batchHolder.value;
// multiple batches
cmIterator = basicCMIRP.find_managed_objects(base_object,
search_control, new String[]{});
isDataPending = cmIterator.next_basic_cm_informations((short)1,
batchHolder);
result = batchHolder.value;
isDataPending = cmIterator.next_basic_cm_informations((short)1,
batchHolder);
result = batchHolder.value;
}
catch (Exception e)
{
Log.error(this, "testFindManagedObjects", "sw error during test
execution", e);
}
```

Notification IRP

NEs generate notifications about network events. Different kinds of events carry different kinds of information such as the following:

- a new alarm, as specified in Alarm IRP: Information Service
- an object creation, as specified in Basic CM IRP: Information Service

The Notification IRP defines an interface through which an IRP manager can subscribe to an IRP agent for receiving a notification.

The 5620 SAM 3GPP OSS interface supports the following notification types:

- alarm—forwarded when alarm events are received from the alarm feeder
- heartbeat—sent by the 3GPP OSS interface to each attached client
- kernel—sent when the 3GPP OSS interface host system notifies the interface about NE creation, deletion or modification

Code 2-20 shows an example of Notification IRP reference retrieval using the Entry Point IRP.

Code 2-20: Notification IRP reference retrieval example

```
StringHolder irp_reference;
irp_reference = new StringHolder();
try
{
entryPointIRP.get_irp_reference(manager_identifier, system_dn,
NotificationIRPImpl.NAME_TO_BIND, irp_reference);
org.omg.CORBA.Object notificationObj =
orb.string_to_object(irp_reference.value);
notificationIRP = NotificationIRPHelper.narrow(notificationObj);
}
catch (Exception e)
{
e.printStackTrace();
}
```

This Notification IRP includes the following methods:

- Subscribe—creates a subscription for receiving notifications
- Unsubscribe—cancels a subscription
- getSubscriptionStatus—obtains the subscription status and informs the IRP agent about whether a subscription is required
- changeSubscriptionFilter—replaces the current subscription filter
- getSubscriptionIds—obtains the ID assigned by the Notification IRP to each valid subscription; a valid subscription is a subscription that is not unsubscribed or removed by the IRP agent
- getNotificationCategories—obtains a list of the notification categories that the IRP agent supports

Subscribe method

Code 2-21 shows the Subscribe method definition. The attach_push call performs the subscription, and returns a subscription ID string.

Code 2-21: Subscribe method

```
      NotificationIRPConstDefs::SubscriptionId attach_push (
         in string manager_reference,
         in ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick,
         in NotificationIRPConstDefs::NotificationCategorySetOpt
            notification_categories,
         in ManagedGenericIRPConstDefs::StringOpt filter
      )
      raises (Attach,
ManagedGenericIRPSystem::ParameterNotSupported,
              ManagedGenericIRPSystem::InvalidParameter,
AlreadySubscribed,
              AtLeastOneNotificationCategoryNotSupported);
```

Code 2-22 shows an example of Subscribe method invocation.

Code 2-22: Subscribe method invocation example

```
public static class SubscriberTest extends SequencePushConsumerPOA
{
.....
}
UnsignedLongOpt time_tick = new UnsignedLongOpt();
time_tick.value(1440);
NotificationCategorySetOpt notification_categories = new
NotificationCategorySetOpt();
notification_categories.value(new String[] {});
StringOpt filter = new StringOpt();
filter.value("");
SubscriberTest subscriber = new SubscriberTest();
org.omg.CORBA.Object subscriberReference = null;
String subscriptionId = null;
org.omg.PortableServer.POA rootpoa =
org.omg.PortableServer.POAHelper.narrow(orb.resolve_initial_referenc
es("RootPOA"));
rootpoa.the_POAManager().activate();
```

```
subscriberReference = rootpoa.servant_to_reference(subscriber);
subscriptionId = orb.object_to_string(subscriberReference);
notificationIRP.attach_push(manager_reference, time_tick,
notification_categories, filter);
```

UnSubscribe method

Code 2-23 shows the UnSubscribe method definition.

Code 2-23: UnSubscribe method

```
      void detach (
         in string manager_reference,
         in NotificationIRPConstDefs::SubscriptionIdOpt
subscription_id
      )
      raises (DetachException,
              ManagedGenericIRPSystem::ParameterNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);
```

Code 2-24 shows an example of UnSubscribe method invocation. The input parameters are the CORBA reference of the subscription and the subscription ID, which are obtained by the subscription operator in the Subscribe method.

Code 2-24: UnSubscribe method invocation example

```
SubscriptionIdOpt sub_id = new SubscriptionIdOpt();
sub_id.value(subscriptionId);
notificationIRP.detach(manager_reference, sub_id);
```

getSubscriptionStatus method

Code 2-25 shows the getSubscriptionStatus method definition.

Code 2-25: getSubscriptionStatus method

```
      NotificationIRPConstDefs::NotificationCategorySet
get_subscription_status
      (
         in NotificationIRPConstDefs::SubscriptionId
subscription_id,
         out ManagedGenericIRPConstDefs::StringOpt filter_in_effect,
         out NotificationIRPConstDefs::SubscriptionStateOpt
subscription_state,
         out ManagedGenericIRPConstDefs::UnsignedLongOpt time_tick
      )
      raises (GetSubscriptionStatus,
              ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);
```

Note — An IRP manager must perform a getSubscriptionStatus operation within the specified timeTick period. Otherwise, the IRP agent removes the subscription.

Code 2-26 shows an example of getSubscriptionStatus method invocation. The input parameters are the CORBA reference of the subscription and the subscription ID, which are obtained by the subscription operator in the Subscribe method.

Code 2-26: getSubscriptionStatus method invocation example

```
StringOptHolder filter_in_effect = new StringOptHolder();
SubscriptionStateOptHolder subscription_state = new
SubscriptionStateOptHolder();
UnsignedLongOptHolder time_tick_holder = new UnsignedLongOptHolder();

String[] cats =
notificationIRP.get_subscription_status(subscriptionId,
filter_in_effect, subscription_state, time_tick_holder);
```

changeSubsciptionFilter method

Code 2-27 shows the changeSubsciptionFilter method definition.

Code 2-27: changeSubsciptionFilter method

```
      void change_subscription_filter (
         in NotificationIRPConstDefs::SubscriptionId
subscription_id,
         in string filter
      )
      raises (ChangeSubscriptionFilter,
              ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);
```

Code 2-28 shows an example of changeSubsciptionFilter method invocation. The input parameters are the CORBA reference of the subscription and the subscription ID, which are obtained by the subscription operator in the Subscribe method.

Code 2-28: changeSubsciptionFilter method invocation example

```
String newFilter = "$.filterable_data(n) == 1";
notificationIRP.change_subscription_filter(subscriptionId,
newFilter);
```

getSubscriptionIds method

Code 2-29 shows the getSubscriptionIds method definition.

Code 2-29: getSubscriptionIds method

```
NotificationIRPConstDefs::SubscriptionIdSet get_subscription_ids (
         in string manager_reference
      )
      raises (GetSubscriptionIds,
              ManagedGenericIRPSystem::OperationNotSupported,
              ManagedGenericIRPSystem::InvalidParameter);
```

Code 2-30 shows an example of getSubscriptionIds method invocation. The input parameters are the CORBA reference of the subscription and the subscription ID, which are obtained by the subscription operator in the Subscribe method.

Code 2-30: getSubscriptionIds method invocation example

```
String[] subscriptions =
notificationIRP.get_subscription_ids(manager_reference);
```

getNotificationCategories method

Code 2-31 shows the getNotificationCategories method definition.

Code 2-31: getNotificationCategories method

```
NotificationIRPConstDefs::NotificationCategorySet
   get_notification_categories (
      out NotificationIRPConstDefs::NotificationTypesSetOpt
         notification_type_list
)
raises (GetNotificationCategories,
        ManagedGenericIRPSystem::OperationNotSupported);
```

Code 2-32 shows an example of getNotificationCategories method invocation. The input parameters are the CORBA reference of the subscription and the subscription ID, which are obtained by the subscription operator in the Subscribe method.

Code 2-32: getNotificationCategories method invocation example

```
NotificationTypesSetOptHolder notification_type_list = new
NotificationTypesSetOptHolder();
String[] ntfCats =
notificationIRP.get_notification_categories(notification_type_list);
```

Communication Surveillance IRP

Ensuring the availability and reliability of the network management solution requires automatic surveillance of the communication between the network manager and the managed system The Communication Surveillance, or CS, IRP, performs this function by sending a periodic heartbeat notification to each active subscriber.

The heartbeat interval is specified by the CNBI.ThreeGPPCORBA.HeartbeatPeriod parameter in the cnbi.properties file. The parameter range is 5 to 60m, and the default value is 5m. An OSS client can retrieve this value through the 3GPP OSS interface, but cannot modify it.

This Communication Surveillance IRP includes the following methods:

- getHeartBeatPeriod

The CS IRP allows an OSS client to detect whether the client subscription is valid. To keep a subscription active, a client must request the subscription status using the getSubscriptionStatus method in the Notification IRP within the time specified by the timeTick attribute in the Subscribe method. If a subscriber does not request the status within this time, the 3GPP OSS interface disables the subscription and stops sending heartbeat notifications to the client. A timeTick value of zero means that the subscription never ends.

The CS IRP is a bidirectional heartbeat mechanism: OSS clients keep subscriptions active by querying the subscription status, and the 3GPP OSS interface periodically sends the heartbeat notification to each active OSS client.

Code 2-33 shows an example of Communication Surveillance IRP reference retrieval using the Entry Point IRP.

Code 2-33: Communication Surveillance IRP reference retrieval example

```
try
{
entryPointIRP.get_irp_reference(manager_identifier, system_dn,
HeartBeatIRPImpl.NAME_TO_BIND, irp_reference);
org.omg.CORBA.Object heartBeatObj =
orb.string_to_object(irp_reference.value);
heartbeatIRP = CSIRPHelper.narrow(heartBeatObj);
}
catch (Exception e)
{
e.printStackTrace();
}
```

getHeartBeatPeriod method

Code 2-34 shows the getHeartBeatPeriod method definition.

Code 2-34: getHeartBeatPeriod method

```
CSIRPConstDefs::Result get_heartbeat_period(
   out CSIRPConstDefs::HeartbeatPeriod heartbeat_period
)
raises (GetHeartbeatPeriod);
```

Code 2-35 shows an example of getHeartBeatPeriod method invocation.

Code 2-35: getHeartBeatPeriod method invocation example

```
ShortHolder heartbeat_period = new ShortHolder();
heartbeatIRP.get_heartbeat_period(heartbeat_period);
```

Kernel IRP

The Kernel IRP contains a method that retrieves the list of network resource IRP versions. It also defines the notification capability for NEs and NE management systems. The 3GPP Kernel IRP standard supports the following notification types:

- object creation
- object deletion
- attribute value change
- requirement to partially or fully resynchronize the configuration information

Note — The 5620 SAM 3GPP OSS interface does not support the resynchronization notification.

This Kernel IRP includes the following methods:

- getNRMIRPVersion

Code 2-36 shows an example of Kernel IRP reference retrieval using the Entry Point IRP.

Code 2-36: Kernel IRP reference retrieval example

```
try
{
entryPointIRP.get_irp_reference(manager_identifier, system_dn,
KernelIRPImpl.NAME_TO_BIND, irp_reference);
org.omg.CORBA.Object kernelObj =
orb.string_to_object(irp_reference.value);
kernelIRP = _KernelCmIrpOperationsHelper.narrow(kernelObj);
}
catch (Exception e)
{
e.printStackTrace();
}
```

getNRMIRPVersion method

Code 2-37 shows the getNRMIRPVersion method definition.

Code 2-37: getNRMIRPVersion method

```
void get_nrm_irp_version
      (
         out ManagedGenericIRPConstDefs::VersionNumberSet
version_number_list,
         out ManagedGenericIRPConstDefs::VersionNumberSet
vse_version_number_list
      )
      raises (GetNRMIRPVersion);
```

Code 2-38 shows an example of getNRMIRPVersion method invocation.

Code 2-38: getNRMIRPVersion method invocation example

```
VersionNumberSetHolder version_number_list = new
VersionNumberSetHolder();
VersionNumberSetHolder vse_version_number_list = new
VersionNumberSetHolder();
kernelIRP.get_nrm_irp_version(version_number_list,
vse_version_number_list);
```

Generic IRP

Each IRP supports a set of common services. The Generic IRP defines the retrieval of IRP profile notification or operation information and the supported versions of a specific IRP. The methods defined in this section are inherited by other IRPs

The Generic IRP includes the following methods:

- getIRPOutline—returns the supported IRP versions for an interface
- getOperationsProfile—returns the operations supported by a specified IRP
- getNotificationProfile—returns the notifications supported by a specified IRP

Note — The returned values vary, depending on the IRP specified.

getIRPOutline method

Code 2-39 shows an example of getIRPOutline method invocation.

Code 2-39: getIRPOutline method invocation example

```
try
{
String irp_version = "";
SupportedIRPListHolder supported_irp_list = new
SupportedIRPListHolder();
Result result =
entryPointIRP.get_irp_outline(irp_version,supported_irp_list);
supported_irp_list.value[0].system_dn);
SupportedIRP supportedIRP = supported_irp_list.value[0];

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

getOperationsProfile method

Code 2-40 shows an example of getOperationsProfile method invocation.

Code 2-40: getOperationsProfile method invocation example

```
try
{
Method[] methods = entryPointIRP
.get_ep_irp_operations_profile(EntryPointIRPImpl.VERSION);
}
catch (Exception e)
{
e.printStackTrace();
}
```

getNotificationsProfile method

Code 2-41 shows an example of getNotificationsProfile method invocation.

Code 2-41: getNotificationsProfile method invocation example

```
try
{
Method[] methods = entryPointIRP
.get_ep_irp_notification_profile(EntryPointIRPImpl.VERSION);
}
catch (Exception e)
{
e.printStackTrace();
}
```

3 — Communicating with OSS clients

3.1 Initial access *3-2*

3.2 Event monitoring *3-2*

3.3 Security *3-5*

3.4 Redundancy *3-8*

3.5 Multiple 3GPP release support *3-8*

3.1 Initial access

In order for OSS applications to connect to the 3GPP OSS interface, the OSS must have access to the list of supported IRPs and their corresponding CORBA references.

The Entry Point IRP provides a standard mechanism for discovering the supported IRPs and the associated CORBA references.

During startup, the 3GPP OSS interface generates the IOR for the Entry Point IRP. This IOR is stored in the following file:

```
$CNBI_HOME/ior/EPIRP.ior
```

The ORB is configured to ensure that this reference does not change when the 5620 SAM restarts.

After an OSS client is given this IOR, it can begin to interact with the 3GPP OSS interface.

Code 3-1 provides an example of how an OSS application can gain initial access to the 3GPP OSS interface.

Code 3-1: Gaining initial access to the 3GPP OSS interface

```
//get entryPoint reference
/input parameter, IOR directory
String iorFile = args[0] + File.separator + "EPIRP.ior";
FileInputStream fis = new FileInputStream(iorFile);
BufferedReader reader = new BufferedReader(new
InputStreamReader(fis));
String ior = reader.readLine();
reader.close();
org.omg.CORBA.Object entryPointObj = orb.string_to_object(ior);
EPIRP entryPointIRP = EPIRPHelper.narrow(entryPointObj);
```

3.2 Event monitoring

This section describe the event monitoring function.

Notification IRP initialization

The 3GPP OSS interface Notification IRP uses the CORBA notification service to send notifications such as alarms, heartbeats, and kernel events to the registered consumers, which are the OSS clients.

When the Notification IRP is initialized, for example, during 5620 SAM main server startup, it creates one event channel and connects a proxy push consumer for that channel with a push supplier. This supplier is used to send all 3GPP notifications.

Subscribing to event notifications

Before an OSS application can receive event notifications, it must do the following, in sequence, as described in Procedure 3-1:

- get the entryPoint reference
- get the notification IRP reference
- create an event consumer and send a subscription request
- wait for events

Procedure 3-1 To subscribe to and unsubscribe from event notifications

Perform this procedure to subscribe to event notifications using the 3GPP OSS interface.

1 Obtain the Entry Point reference using the method shown in Code 3-2.

Code 3-2: getEntryPoint method

```
//get entryPoint reference
/input parameter, IOR directory
String iorFile = args[0] + File.separator + "EPIRP.ior";
FileInputStream fis = new FileInputStream(iorFile);
BufferedReader reader = new BufferedReader(new
InputStreamReader(fis));
String ior = reader.readLine();
reader.close();
org.omg.CORBA.Object entryPointObj = orb.string_to_object(ior);
EPIRP entryPointIRP = EPIRPHelper.narrow(entryPointObj);
```

2 Obtain the Notification IRP reference using the method shown in Code 3-3.

Code 3-3: getNotificationIRP method

```
String manager_identifier = "n.a.";
String system_dn = ""; //Must match the one in cnbi.properties
StringHolder irp_reference;
irp_reference = new StringHolder();
entryPointIRP.get_irp_reference(manager_identifier, system_dn,
"NotificationIRP=1", irp_reference);
org.omg.CORBA.Object notificationObj =
orb.string_to_object(irp_reference.value);
NotificationIRP notificationIRP =
NotificationIRPHelper.narrow(notificationObj);
```

3 Send a subscribe request using the method shown in Code 3-4. The Notification IRP subsequently obtains a proxy push supplier from the event channel and connects the OSS client reference. which is the push consumer, to the proxy.

Code 3-4: attach_push method

```
UnsignedLongOpt time_tick = new UnsignedLongOpt();
time_tick.value(0);
NotificationCwait for eategorySetOpt notification_categories = new
NotificationCategorySetOpt();
```

```
notification_categories.value(new String[] {});
StringOpt filter = new StringOpt();
filter.value("");
SubscriberTest subscriber = new SubscriberTest();
org.omg.PortableServer.POA rootpoa =
org.omg.PortableServer.POAHelper.narrow(orb.resolve_initial_referenc
es("RootPOA"));
rootpoa.the_POAManager().activate();
org.omg.CORBA.Object corbaReference =
rootpoa.servant_to_reference(subscriber);
String manager_reference = orb.object_to_string(corbaReference);
String subscriptionId =
notificationIRP.attach_push(manager_reference, time_tick,
notification_categories, filter);
```

4 Use the 3GPP OSS interface to listen for incoming event notifications that the Notification IRP pushes to the proxy Push Consumer.

5 To unsubscribe from event notifications, use the method shown in Code 3-5.

Code 3-5: detach method

```
//unsubscribe event consumer
SubscriptionIdOpt sub_id = new SubscriptionIdOpt();
sub_id.value(subscriptionId);
notificationIRP.detach(manager_reference, sub_id);
```

Filtering

The 3GPP OSS interface supports a subset of 3GPP filters that can be used in alarm-list and alarm-count retrieval operations. Fault notifications are not filtered.

3GPP alarm filtering supports the following attributes:

- Probable cause
- Perceived severity
- Alarm type
- Ack state

Note — The Probable Cause value INDETERMINATE cannot be used for filtering.

You can combine these attributes using AND and OR operators to create more specific filters.

The supported filtering operations, syntax, and attribute names that can be included as filter criteria are listed in Appendix A.

Connection monitoring

The GPP OSS interface notification service maintains an event queue for each registered consumer. If the number of events in a queue reaches the maximum, for example, if a consumer cannot keep up with the notification rate, the 5620 SAM does the following:

- disconnects the consumer, and marks the consumer subscriptions as invalid
- discards all notifications for the consumer

The 5620 SAM 3GPP OSS interface provides bidirectional connection surveillance using a heartbeat function. If the 5620 SAM main server becomes unavailable, the OSS client requests received on the 3GPP interface fail, and an error message is sent to the OSS client. Subsequent requests succeed if the server becomes available. Contact Alcatel-Lucent technical support to obtain a list of the supported exceptions for a specific method. See chapter 2 for information about connection surveillance.

The 5620 SAM 3GPP OSS interface monitors the state of each client subscription. When a failure is detected, for example, when an exception is raised to the OSS client, the 3GPP OSS interface maintains the OSS client connection and tries to reconnect with the 5620 SAM main server. See section 3.4 for more information about connection failures and redundancy.

Error recovery

The 3GPP OSS interface can recover from system restarts and from client connection interruptions.

3GPP OSS interface restart

After the 3GPP OSS interface restarts, each CORBA reference is updated and the CORBA notification service is restarted, which re-establishes the event channel so that it can accept subscriptions from OSS Clients.

Connection recovery between clients and the interface

The absence of heartbeat notifications, or an invalid status returned by a subscription status query indicates a connection loss between an OSS client and the 3GPP OSS interface. To recover from a connection loss, an OSS client needs to perform the following steps, in sequence:

- Issue an unsubscribe request using the detach method.
- Resubscribe using the attach_push method
- Execute a full resync, which retrieves the alarm list and obtains any missed data.

3.3 Security

This section describes the security considerations for the 5620 SAM 3GPP OSS interface.

User security

Only a 5620 SAM user whose scope of command includes full access permissions on the oss and fm packages can perform 3GPP OSS interface operations. After a 5620 SAM main server installation or upgrade, and before you start the main server, you must add the 3GPP OSS user name and password to the 3GPP OSS configuration. After you start the main server, you must log in as the admin user and create a 5620 SAM 3GPP OSS user account. This account must have the same user name and password as the account defined in the 3GPP OSS configuration. See the *5620 SAM | 5650 CPAM Installation and Upgrade Guide* for 3GPP OSS configuration information. See the *5620 SAM User Guide* for information about creating 5620 SAM user accounts.

SSL security

If SSL is enabled for 5620 SAM main server communication with OSS clients, see the *5620 SAM User Guide* for information on how to enable SSL on the 3GPP OSS interface.

The 3GPP OSS uses a HTTP or HTTPS encryption protocol for sending requests and receiving responses through the 5620 SAM-O server. See the *5620 SAM User Guide* for information on how to configure the 5620 SAM server for HTTP or HTTPS.

Firewall support

The 3GPP OSS interface supports the configuration of a firewall between the 5620 SAM and 3GPP OSS clients. When you configure a firewall, consider the following:

- The firewall must allow inbound connections to at least one port.
- The 3GPP OSS interface must be configured to use the same port that the firewall allows connections to.
- If the inbound port is modified in the firewall rules after the 3GPP OSS interface is installed, you must reconfigure the 3GPP OSS interface, as described in Procedure 3-2.
- The firewall must be configured to allow outbound connections to a port.
- If the OSS client is behind a NAT router, the client can use a VPN with NAT traversal capabilities to reach the 3GPP OSS interface. The VPN is transparent to the 3GPP OSS interface.

Procedure 3-2 To configure a 3GPP OSS interface to operate through a firewall

Perform this procedure to enable OSS client communication with a 3GPP OSS interface through a firewall.

Note — In a redundant 5620 SAM deployment, you must perform this procedure on each 5620 SAM main server in the deployment.

1. Log in to the 5620 SAM main server as the samadmin user.
2. Open the *path*/nms/cnbi/home/config/cnbi.properties file using a plain-text editor

   where *path* is the 5620 SAM main server installation location, typically opt/5620sam/server
3. Locate the following line:

   `CNBI.ThreeGPPCORBA.OrbPort=`
4. Edit the line to read:

   `CNBI.ThreeGPPCORBA.OrbPort=`*`inbound_port`*

   where *inbound_port* is the port value that the OSS clients must use to reach the 3GPP OSS interface
5. Save the file.
6. Close the file.
7. Open a console window on the main server station.
8. Navigate to the following directory:

   path/nms/bin

   where *path* is the 5620 SAM main server installation directory, typically /opt/5620sam/server
9. Enter the following at the prompt:

   `bash$` **`./nmsserver.bash cnbiread_config`** ↵

   The main server loads the updated configuration, and puts the new port value into effect.
10. Close the console window.
11. Log out of the 5620 SAM main server.

3.4 Redundancy

In a redundant 5620 SAM deployment, the 3GPP OSS interface must be installed on the primary and standby main servers. Each 3GPP OSS interface is unaware of the interface on the other main server. See the *5620 SAM | 5650 CPAM Installation and Upgrade Guide* for more information.

After a 5620 SAM server activity switch, the 3GPP OSS interface instance on the former primary main server becomes unresponsive, and heartbeat notifications are not sent to the OSS clients. Each connected OSS client must begin using the entry point IOR of the 3GPP OSS interface instance on the former standby main server, which is the new primary main server, and must subscribe the notification consumers using the Notification IRP of the new primary main server entry point.

Note — In a redundant 5620 SAM deployment, there are two entry point IOR files–one for each 3GPP OSS interface instance that is available to the OSS applications.

3.5 Multiple 3GPP release support

The 3GPP OSS interface can operate in 3GPP R7 or 3GPP R8 compatibility mode, but cannot operate in both modes simultaneously.

An OSS application can switch between the modes by configuring the CNBI.ThreeGPPCORBA.SupportedRelease property in the cnbi.properties file.

The main differences between the two releases are the following:

- the IRP versions
- the notification domain names
- the IRP names

4 — 5620 SAM 3GPP OSS domains

4.1 Fault management *4-2*

4.2 Workflow to obtain an alarm list *4-2*

4.3 Workflow to acknowledge an alarm list *4-2*

4.4 Inventory management *4-2*

4.5 Workflow to obtain a list of NEs *4-6*

4.1 Fault management

A fault management OSS application can use the 5620 SAM 3GPP OSS interface to do the following:

- Receive network event notifications and current alarms using the Notification, Alarm, CS and Kernel IRPs.
- Retrieve alarms from the 5620 SAM using methods from the Alarm IRP.

See chapter 2 for more information about monitoring near-real-time events and alarms.

4.2 Workflow to obtain an alarm list

Perform the following steps to generate a request for listing alarm and event management information. See chapter 2 for a code sample associated with the steps.

1. Get the Entry Point reference.
2. Get the Alarm IRP reference.
3. Retrieve a list of alarms using the get_alarm_list method from the Alarm IRP.

4.3 Workflow to acknowledge an alarm list

Perform the following steps to generate a request to acknowledge alarm and event management information. See chapter 2 for a code sample associated with the steps.

1. Get the Entry Point reference.
2. Get the Alarm IRP reference.
3. Acknowledge a list of alarms using the acknowledge_alarms method from the Alarm IRP.

4.4 Inventory management

The inventory management OSS application has two mechanisms by which it can interface with the 5620 SAM 3GPP OSS interface to retrieve inventory information from LTE devices that are managed by the 5620 SAM:

- Basic inventory available using the CORBA methods defined by the Basic CM IRP.
- 3GPP compliant XML file that is automatically generated and updated daily.

Basic inventory

The 3GPP OSS interface provides the ability to retrieve a list of the LTE-managed NEs. This information acts as auxiliary fault management information, because it is a list of NEs against which alarms can be raised. A limited set of NE attributes is supported.

Note — The 3GPP base object support is limited to the NE level, which means that the 3GPP OSS interface does not recognize the names of objects below the NE level, such as shelves or MDAs.

See chapter 2 for information about the Basic CM IRP for inventory management.

3GPP compliant Inventory File generation and description

The 3GPP OSS Interface can be optionally configured to support the 3GPP Inventory Management IRP (TS 32.692, 32.695). For more information about 3GPP compliant TS, go to http://www.3gpp.org/specifications. Once configured, the 5620 SAM allows for the automatic daily generation of a 3GPP compliant XML Inventory File. This file has inventory data that contains information pertaining to the FRU on hardware, firmware and optional software units of LTE components in a 3G or 4G RAN.

Automatic generation of Inventory File

The generation of the Inventory file is disabled by default when the 3GPP OSS Interface is initially installed. In order to enable the automatic generation of the 3GPP XML Inventory file, the CNBI.SAMO.DATA.START_TIME property located in the *installation_directory*/nms/cnbi/home/config/cnbi.properties file must be configured. The *installation_directory* is the 5620 SAM main server installation directory, typically /opt/5620sam/server

The CNBI.SAMO.DATA.START_TIME property represents the time for the first execution of the inventory data retrieval task. The format expected is *HH:MM*, in 24 h format. Example: 15:35. This property is empty by default.

The following optional properties, also located in the cnbi.properties file, can be configured:

- CNBI.SAMO.DATA.FILENAME—This property is name of the Inventory File. The default value is 3GPP_Inventory_Data_<timestamp>.xml.
- CNBI.SAMO.DATA.RETENTION_PERIOD—This property is the maximum time that the Inventory File will be kept before it is deleted. The minimum and default value is 5 days.
- CNBI.SAMO.DESTINATION.DIRECTORY—This property specifies the destination directory for the Inventory File. The default value is *installation_directory*/lte/inventory.

After configuring the properties described above, you must to execute the following command as the samadmin user in the following directory:

```
installation_directory/nms/bin/nmsserver.bash cnbiread_config
```

Once all the mandatory properties are configured, a 3GPP compliant XML file is generated and placed in the directory specified by the CNBI.SAMO.DESTINATION.DIRECTORY property, and new files will be generated every 24 hours after the configured CNBI.SAMO.DATA.START_TIME. The oldest files will be deleted after the expiration of the time specified by the CNBI.SAMO.DATA.RETENTION_PERIOD property.

Note — It is important that the OSS application and/or system administrator manage the amount of available disk space where these XML files will be stored.

3GPP Inventory File properties description

The 3GPP Inventory file generated by the 3GPP OSS Interface has the following aspects:

- The Inventory File adheres to 3GPP InventoryNRM.xsd schema defined in TS 32.695.
- The root node matches the network domain configured in the 5620 SAM and is specified by the nameOne property in the *installation_directory*/nms/config/nms-server.xmlfile. The default value is 5620sam.
- The next level in the containment is the Managed Elements. The InventoryUnit instances are grouped by the parent Network Element.
- The InventoryUnitId contains the full distinguished name (DN) of the managed object in 3GPP format.

Inventory file attributes

A managed object that contains any of the attributes listed in Table 4-1 will be included in the XML file report. The table indicates whether attributes are mandatory (M) or optional (O).

Table 4-1 Inventory file attributes

Attribute name	Support qualifier
inventoryUnitId	M
inventoryUnitType	M
vendorUnitFamilyType	O
vendorUnitTypeNumber	O
versionNumber	O
vendorName	M
serialNumber	O
dateOfManufacture	O
dateOfLastService	O
unitPosition	O
manufacturerData	O

Note — The optional attributes vendorUnitFamilyType, vendorUnitTypeNumber, and serialNumber in Table 4-1 are mandatory for hardware objects.

Sample 3GPP Inventory File

The sample file displayed below shows inventory data on three managed elements under the 5620sam network domain.

```
<?xml version="1.0" ?>

<InventoryUnit id="SubNetwork=5620sam">

<InventoryUnit  id=
"SubNetwork=5620sam,ManagedElement=138.120.128.126">

<InventoryUnit
id="SubNetwork=5620sam,ManagedElement=138.120.128.126,rack=eNB-1,she
lf=1,cardSlot=2,card=:NA:,BBCard=:NA:">

<attributes>

<inventoryUnitType>MOD: XCEMU</inventoryUnitType>

<vendorUnitFamilyType>BB</vendorUnitFamilyType>

<vendorUnitTypeNumber>300986049</vendorUnitTypeNumber>

<vendorName>ALU </vendorName>

<serialNumber>U80338 </serialNumber>

</attributes>

</InventoryUnit>
```

```
</InventoryUnit>

<InventoryUnit
id="SubNetwork=5620sam,ManagedElement=138.120.148.146">

<InventoryUnit
id="SubNetwork=5620sam,ManagedElement=138.120.148.146,rack=eNB-1,she
lf=1,cardSlot=2,card=:NA:,BBCard=:NA:">

<attributes>

<inventoryUnitType>MOD: XCEMU</inventoryUnitType>

<vendorUnitFamilyType>BB</vendorUnitFamilyType>

<vendorUnitTypeNumber>300986049</vendorUnitTypeNumber>

<vendorName>ALU </vendorName>

<serialNumber>U80338 </serialNumber>

</attributes>

</InventoryUnit>

<InventoryUnit
id="SubNetwork=5620sam,ManagedElement=138.120.148.146,rack=eNB-1,she
lf=1,cardSlot=1,card=:NA:,CBCard=:NA:">

<attributes>

<inventoryUnitType>MOD: XCCM-U BOARD</inventoryUnitType

<vendorUnitFamilyType>CB</vendorUnitFamilyType>

<vendorUnitTypeNumber>0600 </vendorUnitTypeNumber>

<vendorName>ALU </vendorName>

<serialNumber>9U0TS7 </serialNumber>

</attributes>

</InventoryUnit>

</InventoryUnit>

</InventoryUnit>
```


4.5 Workflow to obtain a list of NEs

Perform the following steps to generate a request that returns a list of managed NEs. See chapter 2 for a code sample associated with the steps.

1. Get the Entry Point IRP reference.
2. Get the Basic CM IRP reference.
3. Retrieve a list of NEs using the find_managed_objects method from the Basic CM IRP.

Appendices

A. 5620 SAM-O 3GPP code samples ***A-1***

B. Standards compliance ***B-1***

C. Troubleshooting ***C-1***

A. 5620 SAM-O 3GPP code samples

A.1 Code samples *A-2*

A.1 Code samples

The 5620 SAM-O SDK includes XML interface script samples and 3GPP OSS interface Java code samples.

The library of 3GPP CORBA Java code samples includes fault management and inventory management samples to facilitate rapid OSS application integration with the 5620 SAM-O.

The samples in the 5620 SAM-O SDK library are intended to assist OSS developers with design. For more information about the Java samples that are not included in the 5620 SAM-O SDK library, contact Alcatel-Lucent technical support.

Fault management code samples

Table A-1 lists and describes the fault management code samples that are available for the 3GPP OSS interface.

Table A-1 Fault management Java code samples

Sample Name	Description
CORBA3GPPEventSubscriberSample.java	Sample code to create an event consumer that subscribes to the 3GPP OSS interface and processes events
CORBA3GPPAcknowledgeSample.java	Sample code to acknowledge a list of alarms
CORBA3GPPAlarmListSample.java	Sample code to retrieve a list of alarms

Basic inventory management code samples

Table A-2 lists and describes the basic inventory management code samples that are available for the 3GPP OSS interface.

Table A-2 Basic inventory Java code samples

Sample Name	Description
CORBA3GPPNeListSample.java	Sample code to retrieve a list of NEs

B. Standards compliance

B.1 Standards compliance *B-2*

B.1 Standards compliance

This appendix describes the 5620 SAM 3GPP OSS interface compliance with the 3GPP standards.

The scope of this information is restricted to the following IRPs:

- Entry Point
- Notification
- Communications Surveillance
- Basic CM
- Kernel
- Fault Management

Additionally, the Generic IRP, which is inherited by other IRPs, supports operations to obtain the version, operation, and notification profiles of the other IRPs.

The sources of the IRP information are the 3GPP 32-series R7 specification, published September, 2008, and the 3GPP 32-series R8 specification, published December, 2009.

See the *5620 SAM 3GPP OSS Interface Compliance Statements* document for detailed 3GPP compliance information.

C. Troubleshooting

C.1 3GPP OSS troubleshooting overview *C-2*

C.2 Diagnosing typical 3GPP installation problems on the 5620 SAM server *C-2*

C.3 Useful logs *C-5*

C.1 3GPP OSS troubleshooting overview

The procedures in this appendix describe how to troubleshoot OSS application-specific problems. Use the information contained in this section and the *5620 SAM User Guide* to troubleshoot common client application issues for the 5620 SAM GUI and OSS.

C.2 Diagnosing typical 3GPP installation problems on the 5620 SAM server

You must verify the required platform, environment and license requirements before the 3GPP NBI can be installed. The installation of the 3GPP NBI may fail if the following requirements are not met:

- The 3GPP NBI is supported only on a Solaris x86-based platform. It is not supported on Windows, Linux or SPARC platforms.
- The 3GPP NBI requires a minimum of 12 GB of RAM if installed on a distributed architecture, and a minimum of 24 GB RAM if the SAM server and SAM database are collocated on the same machine. It will report an error if this minimum requirement is not met.
- The 3GPP NBI interface will not install if the minimum Solaris packages for the 3GPP NBI as described in the *5620 SAM | 5650 CPAM Installation and Upgrade Guide* are not installed.
- The 5620 SAM license key must include the license for the 5620 SAM-O.
- The minimum bandwidth requirement between the OSS and 3GPP NBI is 7.1 mbps.

Refer to the *5620 SAM Planning Guide* and *5620 SAM | 5650 CPAM Installation and Upgrade Guide* for more information about platform and environment configurations.

Typical installation problem

It is possible to accidentally skip the enabling of the "Enable 3GPP OSS Interface" option on the Main Server Configuration panel when installing the 5620 SAM server by using using the 5620 SAM/5650 CPAM server installer. You can verify whether the option is enabled by checking if the *installation_directory*/nms/cnbi directory exists. If the directory does not exist, then rerun the 5620 SAM/5650 CPAM server installer and choose Main Server Configuration when asked for the installation type. When you progress to the Main Server Configuration for Clients screen, enable the option for the 3GPP OSS Interface.

3GPP OSS user and password configuration problems

For a brand new installation of the 5620 SAM server, it is possible to forget to edit the cnbi.properties file before adding the CNBI.SAMO.USER and password to the 5620 SAM. If this happens, you must perform Procedure C-1 to update the cnbi.properties file and allow the 3GPP NBI to start successfully.

Note — The password must be MD5-hashed. You can use md5hash.bash utility in the *installation_directory*/nms/bin directory to create the md5hash password.

Procedure C-1 To update the cnbi.properties file in the 5620 SAM

Perform this procedure after configuring the 3GPP NBI user and password properties in the cnbi.properties file.

Note — You will need to perform this procedure when you install the 3GPP NBI on an existing 5620 SAM server.

1. Log into the 5620 SAM main server as the samadmin user.
2. Navigate to the following directory:

   path/nms/bin

   where *path* is the 5620 SAM main server installation directory, typically /opt/5620sam/server
3. Enter the following at the prompt:

   bash$ **./nmsserver.bash cnbiread_config** ↵
4. If the 3GPP NBI application does not come up successfully, you may need to restart the 3GPP NBI processes by performing the following steps.

   i. Enter the following at the prompt:

      bash$ **./nmsserver.bash cnbistop** ↵

   ii. Enter the following at the prompt:

      bash$ **./nmsserver.bash cnbistart** ↵

Troubleshooting 3GPP OSS SSL communication problems

If the 5620 SAM server installation has been configured to support SSL and the OSS cannot communicate with the 3GPP NBI, it is possible that the 3GPP NBI has not been configured for SSL communications. If this is the case, then you must perform Procedure 3-2 to configure the required properties in the *installation_directory*/nms/cnbi/home/config/cnbi.properties file.

Procedure C-2 To verify that the 3GPP OSS Interface is successfully connected to the 5620 SAM server

1. Choose Administration→Security→5620 SAM User Security from the 5620 SAM main menu. The 5620 SAM User Security - Security Management form opens with the General tab displayed.
2. Click on the Sessions tab button.
3. Verify that the CNBI User has a Type of 5620 SAM-O JMS and a State of Active.
4. Click on the Messaging Connections tab button.
5. Verify that the CNBI User has a Client Type of 5620 SAM-O, a State of Connected, and that Durable is checked.
6. Verify the JMS status by performing the following steps:
   - **i** Log in to the 5620 SAM main server as the samadmin user.
   - **ii** Navigate to the following directory:

     path/nms/bin

     where *path* is the 5620 SAM main server installation directory, typically /opt/5620sam/server
   - **iii** Enter the following at the prompt:

     bash$ **./nmsserver.bash jmsstatus** ↵
   - **iv** Verify that the output of the command is similar to the following example:

```
--Subscription: durable? true clientid: cnbi@1288092720429
selector: ALA_clientId in ('cnbi@1288092720429', '') and
((ALA_category = 'FAULT' and MTOSI_NTType = 'NT_ALARM') or
(ALA_category = 'GENERAL' and MTOSI_objectType =
'netw.NetworkElement') or ((MTOSI_objectType =
'StateChangeEvent' and ALA_eventName = 'JmsMi ssedEvents') or
(MTOSI_objectType = 'KeepAliveEvent'))) and mirror is null
and mirrorli is null
```

Verifying that messages are being received by the 3GPP OSS interface from the 5620 SAM

If an OSS is not yet connected and you need to verify that JMS messages are being received by the 3GPP NBI, you should see events being processed in the logs. At least every 30 seconds the JMS KeepAliveEvent messages are being logged in the *installation_directory*/nms/jboss/server/alu_cnm_cnbi/log/cnb.log file similar to the following:

```
.....com.alu.cnm.cnbi.plugin.samo.SAMO_JMSListener.onMessage():
Processing event from SAM-O (START)...
.....com.alu.cnm.cnbi.plugin.samo.SAMO_JMSListener.onMessage:
discarding message with MTOSI object type: KeepAliveEven?..
.....com.alu.cnm.cnbi.plugin.samo.SAMO_JMSListener.onMessage():
Processing event from SAM-O (END), time ms: 1.....
```

Firewall preventing OSS client access to the 3GPP OSS Interface

If a firewall exists between the OSS and the 5620 SAM server, the firewall must be configured to allow outbound connections to any port and allow inbound connections to the CNBI.ThreeGPPCORBA.OrbPort port which is configured in the *installation_directory*/nms/cnbi/home/config/cnbi.properties file. The default value is 9735.

Problems referencing the Entry Point IOR

During initialization, the 3GPP OSS interface generates an IOR for the Entry Point IRP and stores it in the *installation_directory*/nms/cnbi/home/ior/EPIRP.ior file on the 5620 SAM main server. The OSS then uses this IOR to gain initial access to the 3GPP NBI via the Entry Point IRP.

If the OSS is having problems connecting to the 3GPP NBI, one of the first things to verify is whether you retrieved the correct EPIRP.ior from the 5620 SAM server. Copy the IOR from the 5620 SAM Server and compare it with the IOR that the OSS is using.

You can also verify whether the IOR is not corrupt, or whether it has the correct IP address of the SAM server and port by going to http://www2.parc.com/istl/projects/ILU/parseIOR/. Paste the full IOR information, and Parse. It should return the IP address and port on the SAM server.

Incorrect configuration of the System DN

The OSS needs to have defined the System DN. The default System DN is configured in the *installation_directory*/nms/cnbi/home/config/cnbi.properties file as SystemDN? = sam. If this value is different from the default value of "sam" you will receive the following error:

```
*
com.alu.cnm.cnbi.plugin.threegppCORBA.irp.EntryPointIRPImpl.get_irp_
reference(): user= Wrong system_dn parameter: <wrongdn>
```

If you change a value in the cnbi.properties file, you will need to perform the following command after editing the file to update the 5620 SAM:

```
bash$ ./nmsserver.bash cnbiread_config ↵
```

Problems receiving alarms and events when SPAN is configured

If your 5620 SAM server is configured to use SPANs to segment LTE nodes in the network, and you would like to only receive events from nodes in a specific SPAN, the CNBI.SAMO.SPAN.ID property in the cnbi.properties file can be used to configure the Span ID. It allows several SPAN ids separated by commas, as shown in the following example:

```
CNBI.SAMO.SPAN.ID=27, 32
```

C.3 Useful logs

The logs described in this section can provide you with more information to help with debugging problems.

The *installation_director*y/nms/log/cnbi directory contains the following logs:

- cnbi_console.log—provides the latest updates on the status of the 3GPP NBI processes
- cnbi_console.log.<date and time>—several of these exist to provide the historical status

The *installation_directory*/nms/cnbi/home/log directory contains the following logs:

- cnbi-activate_<date and time>.log—reports on the steps to perform the real CNBI application deployment into JBoss application server after the cnbi_install is complete
- cnbi-install_<date and time>.log—reports on the creation of the CNBI home directory structure, preparing the CNBI application deployment to the JBoss application server
- cnbi-pre-activate_<date and time>.log—reports on whether the pre-activation was successful
- cnbi-run-status_twiddle.log—reports on the CNBI status
- cnbi-start_twiddle.log—reports on whether CNBI started

The *installation_directory*/nms/jboss/server/alu_cnm_cnbi/log directory contains the following detailed logs:

- boot.log—provides detailed information on the CNBI activation process
- cnbi.log—provides the most detailed information on all the CNBI processes and communication

The cnbi.log has the following logging or tracing levels:

- ERROR <E>
- WARN <W>
- INFO <I>
- DEBUG <D>
- TRACE <T>

These levels indicate different verbosity and detail level from less to more detailed. ERROR, WARN and INFO trace categories are active by default.

The following is an example of an entry in the cnbi.log file that is marked with <I> for INFO:

```
<2010.10.29 04:59:09 111
-0400><I><rockets><Timer-7><CNBILogger>com.alu.cnm.cnbi.plugin.three
gppCORBA.irp.notification.ConsumersQueueS
izeMonitorTask.handleConsumerStats(): Skipping subscription
CNBI.Notification.Consumer=1288199616340 because is not active.
```

The OSS user information logged with every CORBA action is mapped to the INFO trace category.

Customer documentation and product support

Customer documentation

http://www.alcatel-lucent.com/myaccess

Product manuals and documentation updates are available at 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 Alcatel-Lucent. All rights reserved.

3HE 06509 AAAB TQZZA Edition 01