Oracle Communications Network Charging and Control

Product: OCNCC 4.3
Component: Service Management System
Technical Guide
S'ware version: Release 3.1.2
Guide version: 03.00
Release date: December 2010
Status: Approved
Copyright

Service Management System Technical Guide, Release 3.1.2

03.00

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.
Contents

Chapter 1 .................................................................................................................. 1

System Overview .......................................................................................... 1
Overview ........................................................................................................... 1
What is the Service Management System? ....................................................... 2
Machine Configuration ..................................................................................... 6
Maintaining Network Connections ................................................................. 9
smsTrigDaemon ................................................................................................. 10
Alarms ............................................................................................................... 12
Statistics .......................................................................................................... 15
EDRs ................................................................................................................ 18

Chapter 2 ........................................................................................................... 21
Replication Overview .................................................................................... 21
Overview ......................................................................................................... 21
What is Replication? ...................................................................................... 22
Failover and Error Recovery ......................................................................... 25
Replication in an Unclustered Installation .................................................... 26
replication.def File ......................................................................................... 35
replication.config File .................................................................................. 39

Chapter 3 ........................................................................................................... 41
Replication Check ....................................................................................... 41
Overview ......................................................................................................... 41
Replication Checks ....................................................................................... 42
Database Comparisons ................................................................................ 44
Database Resynchronisations ..................................................................... 46
Auditing .......................................................................................................... 48

Chapter 4 ........................................................................................................... 49
Configuration Management .......................................................................... 49
Overview ......................................................................................................... 49
Configuration Management Editor ............................................................. 50
SMS - Xerlin Integration .............................................................................. 51
Screen and Access Permissions .................................................................. 52
Distributing Configuration Files .................................................................. 53

Chapter 5 ........................................................................................................... 57
Configuring the Environment ......................................................................... 57
Overview ......................................................................................................... 57
Configuration Overview ............................................................................. 58
Configuring the Resource Group in the Clustered Environment .................. 60
Configuring Replication Files ...................................................................... 64
Configuring the Oracle Listener ................................................................. 66
Configuring the SNMP Agent ..................................................................... 69
SMF AlarmMessage Format ........................................................................ 77
Defining the Screen Language ...................................................................... 79
Defining the Help Screen Language .......................................................... 81
Chapter 6 ................................................................. 91

Background Processes on the USMS .................. 91

Overview ................................................................. 91

- cmnConfigRead .................................................. 92
- cmnReceiveFiles ............................................... 93
- smsAlarmDaemon .............................................. 94
- smsAlarmManager ............................................. 97
- smsAlarmRelay .................................................. 99
- smsConfigDaemon ............................................. 102
- smsConfigDaemonScript ................................. 103
- smsCdrProcess.sh ............................................ 105
- smsDbCleanup.sh ............................................. 106
- smsLogCleaner ................................................. 107
- smsMergeDaemon ............................................ 109
- smsMaster ....................................................... 110
- smsNamingServer ........................................... 111
- smsReportsDaemon ......................................... 112
- smsReportScheduler ...................................... 114
- smsReportCleanupStartup.sh ....................... 116
- smsStatsDaemon ............................................. 117
- smsStatsThreshold ......................................... 118
- smsSendConfig.sh ........................................... 119
- smsTaskAgent .................................................. 120
- smsTrigDaemon ............................................... 123

Chapter 7 ............................................................... 129

Background Processes on the UAS .................. 129

Overview ................................................................. 129

- smsApplyConfig.sh ........................................ 130
- cmnPushFiles .................................................. 131
- infMaster ......................................................... 136
- smsAlarmFiles ................................................ 137
- smsLogCleaner ................................................. 139
- smsStatsDaemon ............................................. 141
- updateLoader ..................................................... 150

Chapter 8 ............................................................... 151

Tools and Utilities .............................................. 151

Overview ................................................................. 151

- cmnConfigSyntaxCheck ................................... 152
- compareNode .................................................... 153
- infoDisplayer .................................................... 154
- inputBootstrap ............................................... 155
- repConfigWrite .............................................. 157
- resyncServer .................................................. 158
- smsCompareResyncClient ................................ 159
- smsCompareResyncServer ................................ 162
- smsDumpRepConfig ....................................... 175
- smsLogTest ..................................................... 177
- smsManualRequester ..................................... 178
- smsProcessCdr ............................................... 179
- smsRecordStatistic ........................................ 186

Setting up the screens .............................................. 83
Customising the screens ........................................ 84
Configuring Nodes .............................................. 89
Installing Additional Applications .................... 90
Chapter 9: Reports ................................................................. 189
  Overview ................................................................................. 189
  Reports Database Tables ......................................................... 190
  Installing a Report Script .......................................................... 192
  Report Script Worked Example ............................................... 194
  Database Auditing .................................................................... 199

Chapter 10: Troubleshooting .................................................. 201
  Overview .................................................................................. 201
  Common Troubleshooting Procedures ....................................... 202
  Possible Problems ................................................................. 205
  Index defragmentation .............................................................. 207

Appendix .................................................................................. 209
  Overview .................................................................................. 209
  Glossary of Terms ................................................................. 211
  Index ......................................................................................... 215
About this Document

Scope
The scope of this document includes all the information required to install, configure and administer the USMS application.

Audience
This guide was written primarily for system administrators and persons installing, configuring, implementing and administering the USMS application. The documentation assumes that the person using this guide has a good technical knowledge of the system.

Pre-requisites
Although there are no pre-requisites for using this guide, familiarity with the target platform would be an advantage.

A solid understanding of Unix and a familiarity with IN concepts are an essential pre-requisite for safely using the information contained in this technical guide. Attempting to install, remove, configure or otherwise alter the described system without the appropriate background skills, could cause damage to the system; including temporary or permanent incorrect operation, loss of service, and may render your system beyond recovery.

This manual describes system tasks that should only be carried out by suitably trained operators.

Related documents
The following documents are related to this document:

- SMS User's Guide
- SC3.1 Data Service for OPS/RAC - http://sunsolve.com
- Step-by-Step Installation of 9i RAC on Sun Cluster v3
- Oracle found on http://metalink.oracle.com, Note: 175465.1
- Installing and Configuring Sun Cluster Software
- Xerlin XML editor on http://www.xerlin.org (http://www.xerlin.org/)

Changes in this document
Here are the changes to the document since the last release.

<table>
<thead>
<tr>
<th>Version no.</th>
<th>Revision Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00.00</td>
<td>2 Nov 2007</td>
<td>First draft created split from SMS 3.1.0</td>
</tr>
<tr>
<td>01.00</td>
<td>8 Jan 2008</td>
<td>Final draft published</td>
</tr>
<tr>
<td>02.00</td>
<td>2009-03-24</td>
<td>Calls:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-w option added by 19222.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>documented -g option added by 40321.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>added additional info from cts 28956.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>migrated some parameters to new format.</td>
</tr>
<tr>
<td>03.00</td>
<td>2010-10-11</td>
<td>Calls:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tip added to logjob.conf object by 34064.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note to -h for cmnPushFiles for 28603.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>smsReportsScheduler for 40674.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>incorrect diagram for 108888.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Re-branded.</td>
</tr>
</tbody>
</table>
Document Conventions

**Typographical conventions**

Before you start using this guide, it is important to understand the terms and typographical conventions used in the documentation.

Specialised terms and acronyms are defined in the Glossary at the end of this guide.

<table>
<thead>
<tr>
<th>Formatting convention</th>
<th>Type of information</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Special Bold</strong></td>
<td>Items you must select such as menu options, or names of tabs.</td>
</tr>
<tr>
<td></td>
<td>Emphasis within text.</td>
</tr>
<tr>
<td></td>
<td>Names of database tables and fields.</td>
</tr>
<tr>
<td><strong>Italics</strong></td>
<td>Name of a document, chapter, topic or other publication.</td>
</tr>
<tr>
<td><strong>Button</strong></td>
<td>The name of a button to click or a key to press.</td>
</tr>
<tr>
<td></td>
<td>Example: To close the window, either click <strong>Close</strong> or press <strong>Esc</strong>.</td>
</tr>
<tr>
<td><strong>Key+Key</strong></td>
<td>Key combinations for which the user must press and hold down one key and then press another.</td>
</tr>
<tr>
<td></td>
<td>Example: <strong>Ctrl+P</strong>, or <strong>Alt+F4</strong>.</td>
</tr>
<tr>
<td><strong>Monospace</strong></td>
<td>Text that you must type and examples of code or standard output.</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td>Used to indicate variables or text that should be replaced.</td>
</tr>
<tr>
<td><strong>menu option &gt; menu option &gt;</strong></td>
<td>Used to indicate the cascading menu option to be selected, or the location path of a file.</td>
</tr>
<tr>
<td></td>
<td>Example: <strong>Operator Functions &gt; Report Functions</strong></td>
</tr>
<tr>
<td></td>
<td>Example: <strong>/IN/html/SMS/HelpText/</strong></td>
</tr>
<tr>
<td><strong>hypertext link</strong></td>
<td>Used to indicate a hypertext link on an HTML page.</td>
</tr>
</tbody>
</table>

**Icons**

The following icons are used as visual cues to draw attention to important information.

**Note:** Indicates useful and complementary information. Explanation, comment, or short expansion of the text object that is intended to catch your attention.

**Tip:** Indicates practical but non-essential information that makes the solution easier to use or operate (e.g. keyboard shortcut, alternative way to perform a step in a procedure, etc).

**Warning:** Indicates a caution. If this information is ignored, it could cause possible and irreversible damage to the equipment, data or software.
System Overview

Overview

Introduction

This chapter provides a high-level overview of the application. It explains the basic functionality of the system and lists the main components.

It is not intended to advise on any network or service implications of the product.

In this chapter

This chapter contains the following topics.

What is the Service Management System? .................................................... 2
Machine Configuration .................................................................................... 6
Maintaining Network Connections ................................................................. 9
smsTrigDaemon ............................................................................................ 10
Alarms ........................................................................................................... 12
Statistics ........................................................................................................ 15
EDRs ............................................................................................................. 18
What is the Service Management System?

**Description**

The Service Management System (SMS) product provides service management support for existing Oracle Intelligent Network (IN) products.

The primary function of SMS is to provide operators with access to data used by service logic applications.

SMS provides:

- a central repository for other IN services, such as ACS and CCS, and
- generic functions.

The SMS main menu provides access to all installed services. To access any service, select the item from this menu.

**Functions**

The generic functions of SMS include:

- Security
- Replication
- Statistics gathering
- Alarm Management
- Report generation
- Auditing of Database Changes

Continued on next page
What is the Service Management System?, Continued

Here is an example of the main components of the SMS system.

There are four main subsystems within SMS:

- replication
- configuration management
- reporting functions, and
- file transfer.

Continued on next page
Replication provides the main method of transferring data around the Service Management System. It provides:

- a reliable and fault tolerant delivery of data:
  - from administrators and UASs into the SMF, and
  - changes to persistent data held in the SMF to all relevant UASs (so all parts of the system have consistent data)
- alternative network routing between the UAS and USMS under network failure, or buffered updates under complete network failure or USMS downtime, and
- disaster recovery in the event of a network failure.

Replication moves the following data:

- configuration data for the smsStatsDaemon
- configuration data for other installed IN software (such as ACS, CCS and UBE)
- any update of application data due to the actions of the service running on the UASs (including client account and call routing data)
- system, application and interface statistics, and
- alarms.

For more information about replication, see Replication Overview (on page 21).

The Configuration Management editor provides a graphical interface to edit and centrally control configuration data, prior to being deployed to the eserv.config data saved on other platforms.

It provides a reliable mechanism for managing configuration data for all NCC products. It effectively eliminates problems encountered while using a text-based editor, such as broken configurations and inconsistent data.

The Reporting functions enable the administrator to run reports against the data collected in the SMF.

The Reports are configured in the SMS Java Administration screens.

There are two main methods of data transfer:

- replication, and
- file transfer (using ftp).

This table describes the main components in SMS.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsMaster</td>
<td>Receives update requests and forwards them to the SMF.</td>
<td>smsMaster (on page 110)</td>
</tr>
<tr>
<td>SMF</td>
<td>The main SMF on the USMS.</td>
<td></td>
</tr>
<tr>
<td>UAS</td>
<td>The databases on the UASs. They hold a subset of the data on the SMF.</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
What is the Service Management System?, Continued

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateLoaders</td>
<td>Receives update orders from the smsMaster and inserts them into the UASs.</td>
<td><em>updateLoader</em> (on page 150)</td>
</tr>
<tr>
<td>Update Requesters</td>
<td>Update requesters run on box_SMP_ac&gt;s and UASs and may run on other IPs as well. They send update requests to the smsMaster. They include the smsAlarmDaemon and the smsStatsDaemon.</td>
<td></td>
</tr>
<tr>
<td>smsTaskAgent</td>
<td>Forwards administrator’s instructions to the smsMaster.</td>
<td><em>smsTaskAgent</em> (on page 120)</td>
</tr>
<tr>
<td>smsAlarmDaemon</td>
<td>Collects alarms from local sources and forwards them to the smsMaster.</td>
<td><em>Alarms</em> (on page 12)</td>
</tr>
<tr>
<td>smsAlarmRelay</td>
<td>Monitors the alarms in the SMF and forwards alarms to administrators.</td>
<td><em>Alarms</em> (on page 12)</td>
</tr>
<tr>
<td>smsReportsDaemon</td>
<td>Enables the user to run reports against the data held in the SMF.</td>
<td><em>smsReportsDaemon</em> (on page 112)</td>
</tr>
<tr>
<td>smsStatsDaemon</td>
<td>Collects statistics and forwards them to the smsMaster.</td>
<td><em>Statistics</em> (on page 15)</td>
</tr>
</tbody>
</table>
Machine Configuration

There are three configurations that SMS can be installed on. They are:

- on a single machine
- with one USMS on one machine and one or more UASs on separate machines, and
- with multiple USMSs connected to a raid array and one or more UASs on separate machines.

Using the unclustered machine configuration, the smsSms package is installed on the USMS machine. The smsScp package is installed on one or more UASs.

This configuration provides resilience by using a failover system from the USMS to the UASs. However, while the USMS is unavailable, no configuration updates can be forwarded to the UASs.

Here is an example of replication in an unclustered installation.
Here is an example of the main components of a clustered USMS system.

Using the single machine configuration, all required SMS functionality is installed on a single machine. Because all SMS functionality is on a single machine the parts of SMS which are involved in connecting the different components are removed.

This results in a simple, easy to administer system. However, because the system runs on one machine, resilience is reduced.
Here is an example of the components in SMS installed on a single machine.
# Maintaining Network Connections

**Introduction**

All replication elements (nodes) establish TCP connections with a Master Replicator by implicitly connecting to one.

To maintain reliable connection between nodes of the replication system two methods are employed to strengthen the underlying TCP protocol to be used:

- Heartbeating
- Dual network connection

**Heartbeating**

A simple heartbeating mechanism is used to overcome TCP's failure to detect connection severance (for example, cable failure).

Every node connected to a Superior Master node sends a periodic heartbeat message to which the master will respond with an acknowledgement. This ensures both ends of the connection can detect failure within one heartbeat period.

Whenever a connection fails the connecting element should attempt to reconnect to a Superior Master. If the Superior Master is part of a cluster, the connecting element will attempt to connect to the next Master in the cluster.

**Dual Network Connection**

The replication system supports dual network connection to overcome a potential single point of failure (that of the underlying transport medium).

Each node has the ability to have two addresses by which it can be reached; a primary and a secondary address. These addresses can (and should) be on separate networks.

Whenever a connection to a Superior Master is required by an element, two connection attempts are made:

1. Primary address (over the primary network) and,
2. Secondary address (over a secondary network).

The replication element will use the first connection to succeed, closing the other connection first.

If required, a configurable delay (of up to one second) will occur between the connection attempt to the primary address and the secondary one.

This provides the ability to favour the primary network over the secondary (for example, if one network has a better known latency).

If no delay is configured the connection attempts will be attempted simultaneously, if both networks have similar latency which one ultimately gets used will be unpredictable.
smsTrigDaemon

**Purpose**
smsTrigDaemon manages control plan execution requests. It runs on the USMS platform.

smsTrigDaemon accepts control plan execution requests from either a remote PI client or the Java management screens. It forwards requests to ACS through the xmlTcapInterface on the UAS platform. An indication of whether or not the requests were successful passes back from the ACS to the initiating client.

**Architectural overview**
The following diagram shows smsTrigDaemon and components that surround it.

![Diagram showing smsTrigDaemon and components](image)

**Message flows**
This table describes message flows that smsTrigDaemon uses.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Java management screens send control plan execution requests to smsTrigDaemon over a CORBA transport layer. Each request contains the name of the control plan to be executed, the UAS service handle, the CLI of the subscriber against which the control plan should be executed, an optional called party number and extensions.</td>
</tr>
<tr>
<td>2</td>
<td>A remote PI client sends control plan execution requests to the provisioning interface. As with stage 1, each request contains the name of the control plan to be executed, the UAS service handle, the CLI of the subscriber against which the control plan should be executed, an optional called party number and extensions.</td>
</tr>
</tbody>
</table>

*Continued on next page*
smsTrigDaemon, Continued

Message flows (continued)

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>The provisioning interface forwards requests to smsTrigDaemon over the FIFO layer transport layer.</td>
</tr>
<tr>
<td>4</td>
<td>Using an XML request, smsTrigDaemon forwards the control plan execution request to the xmlTcapInterface on the UAS platform.</td>
</tr>
</tbody>
</table>
| 5     | The xmlTcapInterface constructs an InitialDP and sends it to ACS through the SLEE.  

For more information about the SLEE, see SLEE Technical Guide. |
| 6     | An indication of success or failure is returned to the xmlTcapInterface using a Connect, Continue or ReleaseCall component. |
| 7     | The indication of success or failure is sent to smsTrigDaemon using an HTTP response. smsTrigDaemon then sends the indication back to the client.  

Note: Third parties can also send XML requests directly to the xmlTcapInterface. |

Components

The smsTrigDaemon interacts with three subsystems.

- Provisioning Interface
- xmlTcapInterface, and
- SLEE.

PI

The Provisioning Interface (PI) provides a mechanism for manipulating data in the SMF. It enables bulk or scripted operations on SMF data where manual input using the Java management screens would be inefficient.

For more information, refer to PI Technical Guide.

xmlTcapInterface

The xmlTcapInterface enables the SLEE to inter-work with a TCAP protocol. The interface converts XML messages arriving from smsTrigDaemon into SLEE events. Similarly, the interface converts events arriving from the SLEE into XML messages that smsTrigDaemon understands.

For more information, refer to TCAP Interfaces Technical Guide.
Alarms

Introduction

Alarms from the USMS and UASs are collected in the SMF using replication. A set of tools enable management of the alarms in the SMF. Functions include:

- filtering alarms
- setting notification destinations, and
- monitoring.

This functionality is configured using the alarms screens in SMS. For more information about configuring alarms, see the SMS User’s Guide.

Alarms can be generated from monitoring statistics.

Alarms diagram

Here is an example of the alarms transfer process.
This table describes the stages involved in collecting and reporting about alarms within the SMS system using replication.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | Alarms are collected by the smsAlarmDaemon on the USMS and UASs. Sources include:  
- the syslog file, and  
- Oracle logs. |
| 2     | The smsAlarmDaemon sends an update request to the Superior Master (usually the smsMaster).  
**Exception:** The smsAlarmDaemon on the USMS makes its updates directly to the SMF, without sending anything to the smsMaster. |
| 3     | When the Superior Master receives an update request, it will insert the updated data into the SMF_ALARMS_MESSAGE table of the SMF. |
| 4     | The smsAlarmManager matches each alarm instance in the SMF_ALARMS_MESSAGE table with the correct alarm type from the SMF_ALARM_DEFN table, and additional information about the alarm type is saved with the alarm instance. |
| 5     | The smsAlarmRelay process will monitor the SMF_ALARMS_MESSAGE table and forward alarms to the specified external resource. |

**Note:** The Administrator can run reports on the collected alarms using the reports screens in SMS (which are executed by the smsReportsDaemon).

Alarms can be generated from specific statistical measures. The smsStatsThreshold process monitors the SMF_STATISTICS table in the SMF database. When a statistic or statistics match a rule specified in the SMF_THRESHOLD_DEFN table, the smsStatsThreshold process inserts an alarm record into the SMF_ALARM_MESSAGE table in the SMF database.

For more information about configuring statistics thresholds, see the *SMS User’s Guide*.

Enhanced Fault Management takes the alarms that are produced by the system and matches alarm instances to information that is held in the database for each alarm type. The alarm instances, including the additional information can then be relayed to an external resource for further processing.

This table describes the roles of the components involved in the alarms process.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsAlarmDaemon</td>
<td>Collects alarms from local sources and forwards them to the smsMaster.</td>
<td><em>smsAlarmDaemon</em> (on page 94)</td>
</tr>
<tr>
<td>smsMaster</td>
<td>Receives alarms from smsAlarmDaemons and forwards them to the SMF.</td>
<td><em>smsMaster</em> (on page 110)</td>
</tr>
</tbody>
</table>
## Alarms, Continued

### Description of processes and executables (continued)

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsAlarmRelay</td>
<td>Monitors the SMF_ALARM_MESSAGE table in the SMF and forwards alarms to relevant notification points (including SNMP).</td>
<td>smsAlarmRelay (on page 110)</td>
</tr>
<tr>
<td>smsReportsDaemon</td>
<td>Enables the user to run reports against the alarms held in the SMF.</td>
<td>smsReportsDaemon (on page 112)</td>
</tr>
<tr>
<td>smsStatsThreshold</td>
<td>Monitors the SMF_STATISTICS table in the SMF. If the statistics meet certain rules, the this process will create an alarm and insert it into the SMF_ALARM_MESSAGE table in the SMF.</td>
<td>smsStatsThreshold (on page 118)</td>
</tr>
<tr>
<td>smsAlarmManager</td>
<td>The smsAlarmManager matches alarm instances with the alarm definitions stored in the SMF_ALARM_DEFN table on the SMF, and adds the extra information stored in the definition to each instance of that alarm as it occurs.</td>
<td>smsAlarmManager (on page 97)</td>
</tr>
</tbody>
</table>

### Alarm replication and buffering

The smsAlarmDaemon filters alarms before they are sent. This enables:

- protection against the USMS being flooded with alarms, and
- filtering of repeating alarms.

For more information about buffering alarms, see smsAlarmDaemon (on page 94).
Statistics

Statistics generated by the USMS and UASs are collected in the SMF_STATISTICS table of the SMF database. A set of tools provides management functions. Functions include:

- filtering statistics
- setting rules for statistics thresholds which raise alarms, and
- running reports against the statistics held in SMF_STATISTICS.

For more information about using these functions, see the SMS User's Guide.

Here is an example of the statistics collection process.
This table describes the roles of the components involved in the statistics process.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsStatsDaemon</td>
<td>Collects statistics from UASs and forwards them to smsMaster.</td>
<td>smsStatsDaemon (on page 141).</td>
</tr>
<tr>
<td>smsMaster</td>
<td>Receives statistics from the smsStatsDaemons and forwards them to the smsMaster for insertion into the SMF.</td>
<td>smsMaster (on page 110).</td>
</tr>
<tr>
<td>smsReportsDaemon</td>
<td>Enables the user to run reports against the statistics held in the SMF.</td>
<td>smsReportsDaemon (on page 112).</td>
</tr>
<tr>
<td>smsStatsThreshold</td>
<td>Monitors the SMF_STATISTICS table in the SMF. If the statistics meet certain rules, the smsStatsThreshold process will create an alarm and forward it to the smsAlarmDaemon on the USMS.</td>
<td>smsStatsThreshold (on page 118).</td>
</tr>
</tbody>
</table>

This table describes the stages involved in collecting statistics within the SMS system using replication.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Statistics are gathered by the statistics daemon process (smsStatsDaemon) which runs on each UAS. Statistics which are collected include: &lt;br&gt;• statistics from the shared memory which are generated by the slee_acs &lt;br&gt;• TCAP statistics from files saved by the TCAP interface, and &lt;br&gt;• system statistics from the kernel.</td>
</tr>
<tr>
<td>2</td>
<td>At regular intervals, the smsStatsDaemon sends the values to the smsMaster process on the &lt;box_SMP_ac&gt; as an update request.</td>
</tr>
<tr>
<td>3</td>
<td>The smsMaster adds the new statistics to the SMF_STATISTICS table in the SMF.</td>
</tr>
<tr>
<td>4</td>
<td>The Administrator can run reports on the collected statistics using the statistics screens in SMS (which are executed by the smsReportsDaemon).</td>
</tr>
</tbody>
</table>

Alarms can be generated from specific statistical measures.

The smsStatsThreshold process monitors the SMF_STATISTICS table in the SMF database. When a statistic or statistics match a rule specified in the SMF_THRESHOLD_DEFN table, the smsStatsThreshold process inserts an alarm record into the SMF_ALARM_MESSAGE table in the SMF database.

For more information about configuring statistics thresholds, see the SMS User’s Guide.

Continued on next page
The statistics system can collect any SMS-compatible IN application statistics. These are typically coarse values related to the general performance and behaviour of the application. Typical statistics values include:

- Total number of requests from SSF
- Number of call instances resulting in error treatment
- Number of calls from invalid geographical locations
- Number of calls reaching successful call completion to international locations
- Number of calls reaching successful call completion to international category

Statistics sources may include:

- system statistics from the syslog
- system statistics from the operating system
- statistics from the Sigtran stack
- statistics from shared memory.

Note: For statistics about call processing, see also the ACS Technical Guide.
**EDRs**

**Introduction**

The SMS product provides a complete, integrated reporting mechanism for Event Data Records. It allows the developers of SMS-compatible IN applications to add report functions to their product, via the SMS reports interface.

**EDR file transfer diagram**

Here is an example of the transfer of files between UASs and the USMS.

This table describes the stages involved in transferring files around the system using the Common File Transfer process. The files usually transferred are EDRs.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On the UASs, cmnPushFiles collects files from the configured input directory and transfers them to the configured output directory on the USMS through an stdout. It adds the destination directory to the file.</td>
</tr>
<tr>
<td>2</td>
<td>If there is a failure in the transfer, cmnPushFiles will copy the files to the configured retry directory to attempt the transfer again later.</td>
</tr>
<tr>
<td>3</td>
<td>When the files have been successfully transferred to the SMS, cmnPushFiles moves the files to the configured completed directory.</td>
</tr>
<tr>
<td>4</td>
<td>On the USMS, cmnReceiveFiles scans the configured input directory and moves any files to the directory specified in the file.</td>
</tr>
<tr>
<td>5</td>
<td>smsCdrProcess.sh scans its input directory for *.cdr files and moves them to its processed directory.</td>
</tr>
</tbody>
</table>

*Continued on next page*
EDRs, Continued

### Description of processes and executables

This table describes the roles of the components involved in the alarms process.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmnPushFiles</td>
<td>Reads files from a specified directory and transfers them to the SMS via stdout. Depending on the success of the transfer, the file will also be moved to another directory on the origination UAS.</td>
<td>cmnPushFiles (on page 131).</td>
</tr>
<tr>
<td>cmnReceiveFiles</td>
<td>Collects files from the input directory on the SMS and writes them to the specified output directory on the SMS.</td>
<td>cmnReceiveFiles (on page 93).</td>
</tr>
<tr>
<td>smsCdrProcess.sh</td>
<td>Provides a set of EDR processing and archiving functions.</td>
<td>smsCdrProcess.sh (on page 105).</td>
</tr>
<tr>
<td>smsReportDaemon</td>
<td>Enables the user to run reports against the statistics held in the SMF.</td>
<td>smsReportsDaemon (on page 112).</td>
</tr>
</tbody>
</table>

### Directory structure and filenames

So that the Unix transfer scripts can locate the output EDR file, the file should be named according to the naming convention. This is usually done by the processes which create the files.

The directory structure which holds the files is in:

/IN/service_packages/SMS/cdr/

For more information about the directory structure, see the ACS Technical Guide.

The file name is `<Application-ID>.cdr`. In this case, the complete specification of the currently active EDR filename for the ACS application will be:

APP_yyyymmdhhmmss.txt

Where:

- APP is the three letter acronym for the originating process, and
- yyyymmdhhmmss is the date and time the file started to be written to.

There is no need for the application to provide any further detail in the file name, as the subsequent processing of the EDR files can perform this. The file names for archived files on the UAS and USMS are detailed in the section that deals with the subsequent processing of these files.

### EDR intermediate file format

The intermediate EDR, as output from the SMS EDR API is written to the following directory:

/IN/service_packages/SMS/cdr/current/

The format of the file is a | separated list of TAG=VALUE pairs, except for the first entry which is the service name followed by a |. Each record is new line separated.

**Example:**

`Continued on next page`
## EDRs, Continued

### EDR intermediate file format (continued)

<table>
<thead>
<tr>
<th>Event Type</th>
<th>SN</th>
<th>TN</th>
<th>CGN</th>
<th>TCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acs_Service</td>
<td>1800906420</td>
<td>4770360</td>
<td>9380360</td>
<td>1999060312449</td>
</tr>
<tr>
<td>Acs_Service</td>
<td>1800906421</td>
<td>4770361</td>
<td>9380361</td>
<td>1999060312450</td>
</tr>
<tr>
<td>Acs_Service</td>
<td>1800906422</td>
<td>4770362</td>
<td>9380362</td>
<td>1999060312457</td>
</tr>
<tr>
<td>Acs_Service</td>
<td>1800906423</td>
<td>4770363</td>
<td>9380363</td>
<td>1999060312521</td>
</tr>
<tr>
<td>Acs_Service</td>
<td>1800906424</td>
<td>4770364</td>
<td>9380364</td>
<td>1999060312590</td>
</tr>
<tr>
<td>Acs_Service</td>
<td>1800906425</td>
<td>4770365</td>
<td>9380365</td>
<td>1999060312449</td>
</tr>
<tr>
<td>Acs_Service</td>
<td>1800906426</td>
<td>4770366</td>
<td>9380366</td>
<td>1999060312449</td>
</tr>
<tr>
<td>Acs_Service</td>
<td>1800906427</td>
<td>4770367</td>
<td>9380367</td>
<td>1999060313036</td>
</tr>
<tr>
<td>Acs_Service</td>
<td>1800906428</td>
<td>4770368</td>
<td>9380368</td>
<td>1999060312036</td>
</tr>
</tbody>
</table>
# Replication Overview

## Overview

<table>
<thead>
<tr>
<th>Introduction</th>
<th>This chapter explains the replication system used in USMS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>In this chapter</td>
<td>This chapter contains the following topics.</td>
</tr>
<tr>
<td>What is Replication?</td>
<td>.................................................................22</td>
</tr>
<tr>
<td>Failover and Error Recovery</td>
<td>.................................................................25</td>
</tr>
<tr>
<td>Replication in an Unclustered Installation</td>
<td>.................................................................26</td>
</tr>
<tr>
<td>replication.def File</td>
<td>.................................................................35</td>
</tr>
<tr>
<td>replication.config File</td>
<td>.................................................................39</td>
</tr>
</tbody>
</table>
What is Replication?

Introduction

Replication is the system which transfers data between nodes in the IN installation.

Data flow

The SMF database on the USMS holds the full set of authoritative data within the system. Data required for call processing and resilience is forwarded to the UASs on the UASs using SMS replication. Updates are received from processes on the USMSs and the UASs and from the SMS administration screens.

This table describes the stages involved in replicating data around the system.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | Update Requests may come from either:  
|       | • the administration screens, or  
|       | • an event on the USMS or UASs. |
| 2     | If the update comes from the administration screens, it will either:  
|       | • be forwarded to the smsTaskAgent, and then through to an  
|       | • smsMaster, or  
|       | • be inserted directly into the SMF database.  
|       | If the update request comes from the USMS or UASs, the relevant  
|       | update requester will send an update request to an smsMaster (parent). |
| 3     | When an smsMaster (parent) receives an update request, it will:  
|       | 1 send an update order to all configured destination replication  
|       | • groups (there may be no relevant groups, in which case no order  
|       | • will be sent), and  
|       | 2 spawn a local smsMaster (child) process to insert the updated data  
|       | • into the SMF database. |
| 4     | The updateLoader on the relevant UASs will read the update order from  
|       | • the socket and insert the data into the UAS database. |
| 5     | If requested to do so, the updateLoader will send a confirmation that  
|       | • the update has been successfully completed to the smsMaster. |

Nodes

Replication occurs between nodes in the system. Nodes allow specific processes on machines to be replicated to and from, and for more than one node to exist on a single machine. Each node has a node number which identifies the node.

For more information about configuring nodes, see the SMS User’s Guide.

Continued on next page
### What is Replication?, Continued

<table>
<thead>
<tr>
<th>Superior Master nodes</th>
<th>Superior Master nodes are forwarded all data update requests within SMS, and distribute update orders to all UASs that require the replication data via the updateLoaders.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>In a clustered installation, the Superior Master role is shared between the available smsMaster nodes on the USMSs.</td>
</tr>
<tr>
<td></td>
<td>In an unclustered installation, the Superior Master node is the node with the highest node number in the system. This will usually be the smsMaster on the USMS, but at times may be an infMaster on an UAS.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Update Loader nodes</th>
<th>Update loader nodes will run on any UAS that requires database updates. They are the updateLoader processes running on the UASs. They accept update orders from Superior Master nodes and insert the data into the local UAS database.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The Update Loaders on a single UAS platform are independent of each other and are treated as separate replication nodes to the replication system. Hence there can be more than one per machine, although in practice there is normally just one.</td>
</tr>
<tr>
<td></td>
<td>An Update Loader must always be connected to a Master, even if it is not receiving any information from the Master it will have a connection.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Update Requester nodes</th>
<th>Update Requesters create update requests in response to specific events on the UASs and send them to the Superior Master to update the centralised data (and from there it will be replicated out to the relevant UASs). Update Requesters include:</th>
</tr>
</thead>
</table>
|                        | - replicationIF  
|                        | - smsAlarmsDaemons, and  
|                        | - smsStatsDaemons. |
|                        | Update requesters do not need to be configured in the database. |

<table>
<thead>
<tr>
<th>Replication groups</th>
<th>A replication table has one or more replication groups. A replication group can be assigned to one or more replication nodes.</th>
</tr>
</thead>
</table>
| Example:           | - Replication Group A resides on Node 1, Node 2 and Node 3  
|                     | - Replication Group B resides on Node 1 and Node 3 |

<table>
<thead>
<tr>
<th>Primary replication nodes</th>
<th>Primary nodes can be defined for a specific replication group. The primary is the highest priority destination node for the data defined in the replication group. This enables the IN to assign particular services to specific nodes, but still provide a failover to other nodes as required.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This only sets the node as the primary for the specific group involved and is independent of other groups. A node may be defined as a primary for one group without being a primary for another group.</td>
</tr>
</tbody>
</table>
| Example:                  | - Replication Group A resides on Node 1, Node 2 and Node 3, where Node 3 is the primary for group A.  
|                           | - Replication Group B resides on Node 1 and Node 3, where Node 1 is the primary for group B. |

*Continued on next page*
Primary replication nodes (continued)

Primary nodes are not required unless a service will be running with different priority on different nodes.

Update requests to primary nodes

Primary node status is relevant for processes which are requesting an smsMaster to update the SMF.

The update processes have three types of Update Requests:

1. Make the change and do not confirm that it has been made.
2. Send a notification when the change has been made to the SMF.
3. Send a notification when the change has been made to the primary replication node for this replication group.

The primary node status is used when the third type of update request is used. While the update may be successful without the primary node being configured, the requesting process may register errors if the notification of the update is not received.

For more information about setting Primary and Secondary status within a replication group, see SMS User's Guide.

Master Controllers

A Master Controller is any process which provides instructions to a Superior Master node. Possible instructions include:

- update configuration
- merge databases, and
- resync databases.

Master Controllers include executables started from the command line and functions embedded in other processes. They include:

- smsTaskAgent
- resyncServer
- smsCompareResyncServer
## Failover and Error Recovery

### Introduction

If a node becomes unavailable for any reason, the system will attempt to continue functioning. The nodes which remain available will continue to operate normally. Updates for the node which is unavailable will be queued for as long as the queue space lasts.

When the node becomes available again, the queued updates will be resent.

If nodes become out of sync to the point where they will not automatically recover, a manual resync can be run.

### updateLoader failure

If the Update Loader fails, then the updates are queued until it is back on-line. If the Update Loader is still down after a period of time and a smsMaster’s Pending Queue reaches its configured maximum size, then the Update Loader is marked as “Out Of History” by that smsMaster and its updates are removed. If this happens, after the Update Loader is back on-line, a total database re-synchronisation is performed with the smsMaster.

### Update queuing

If the nodes become disconnected, a number of processes will queue updates until the connection is restored. Once the connection is restored, the queued updates will execute normally.

The smsMaster queues all updates it sends out until an acknowledgement is sent out by the receiving updateLoader. The number updates that will be queued is set in the smsMaster configuration.

The updateLoader will queue all uncompleted updates in the following file:

<updateLoader node number>-queuedOrders.dat

### Further information

For more information on failover and error recovery processes, see Replication Check (on page 41).
Replication in an Unclustered Installation

Here is an example of replication in an unclustered installation.

This table describes the components of replication in an unclustered installation.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsMaster</td>
<td>Runs on the USMS handling updates throughout SMS. This is the Superior Master for all connected nodes.</td>
<td>smsMaster (on page 110)</td>
</tr>
<tr>
<td>infMaster</td>
<td>An infMaster runs on each UAS. If it becomes the node with the highest number of all connected nodes, it will stand in as the Superior Master until a higher node number becomes available again.</td>
<td>infMaster (on page 136)</td>
</tr>
</tbody>
</table>

Continued on next page
Replication Components (continued)

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>updateLoaders</td>
<td>An updateLoader runs on each UAS. It manages all incoming update orders and inserts updated data into the UAS. At any point in time, an updateLoader will be connected to a specific Superior Master.</td>
<td><em>updateLoader</em> (on page 150)</td>
</tr>
<tr>
<td>update requesters</td>
<td>update requesters may run on any machine. They send update requests to the Superior Master.</td>
<td><em>Update Requester nodes</em> (on page 23)</td>
</tr>
<tr>
<td>smsMergeDaemon</td>
<td>The smsMergeDaemon runs on the USMS and monitors the connections between the USMS and the UASs. If it notices a break in the connection, it may start a merge to update the disconnected nodes.</td>
<td><em>smsMergeDaemon</em> (on page 109)</td>
</tr>
<tr>
<td>smsTaskAgent</td>
<td>The smsTaskAgent accepts instructions from the SMS Administration screens and produces instructions for the smsMaster. It will generate the replication config file and copy it to the UASs</td>
<td><em>smsTaskAgent</em> (on page 120)</td>
</tr>
<tr>
<td>smsNamingServer</td>
<td>The smsNamingServer enables non-SMS components to connect to elements within the SMS.</td>
<td><em>smsNamingServer</em> (on page 111)</td>
</tr>
<tr>
<td>SMF</td>
<td>This Oracle database holds authoritative data for all UASs.</td>
<td></td>
</tr>
<tr>
<td>UASs</td>
<td>These Oracle databases hold the subset of SMF data required to route calls.</td>
<td></td>
</tr>
</tbody>
</table>

Updates

The replication system performs ‘row’ level updates and buffers updates to reduce processing load on the real-time system elements. This is achieved by holding the update requests in a memory resident queue (called the Pending Updates Queue) until replication has been successfully completed.

Update requests are performed in the order they arrive at the Superior Master.

Inferior Master nodes

An Inferior Master node is a Master node with a node number less than the current Superior Master node’s node number. It will not perform any function unless it becomes the available Master node with the highest node number (in which case it will become the Superior Master).

*Continued on next page*
## Replication in an Unclustered Installation, Continued

### Node numbers

This table lists the node number ranges and their details for an unclustered installation.

<table>
<thead>
<tr>
<th>Node Numbers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>This node number must be assigned to the smsMaster process on the USMS.</td>
</tr>
<tr>
<td>17-255</td>
<td>These node numbers are available to infMaster processes on the UASs.</td>
</tr>
<tr>
<td>256-511</td>
<td>These node numbers are available to updateLoaders on the UASs. Usually they start from node number 301.</td>
</tr>
</tbody>
</table>
| 512-999      | These node numbers are available to updateRequesters. Usually they will be configured in the following pattern:  
  - 601-699  Replication IF nodes  
  - 701-799  smsStatsDaemon nodes  
  - 801-899  smsAlarmDaemon nodes  
| 1000         | In an unclustered installation, this node number is used for the smsMergeDaemon. |

**Note:** Node numbers are unique.

### Failover

If a node becomes disconnected from the smsMaster node (due to network failure or a problem with the USMS), it will attempt to contact the other nodes in descending node number order until it locates a node it can connect to.

An infMaster on one of the UASs will become the acting Superior Master until the failure is resolved. Once the smsMaster becomes available again, the smsMergeDaemon will instruct the infMaster to merge its updates with the smsMaster.

If the infMaster which is the acting Superior Master becomes unavailable before the smsMaster is available again, the infMaster with the next node number will be used instead.

*Continued on next page*
Here is an example showing all nodes in an unclustered configuration connected to the smsMergeDaemon.
The following diagram depicts an isolated UAS in an unclustered environment.

Where an UAS has been isolated from the Master it looks for and connects to the master in the network, which has the next lowest node number. In the diagram above, UAS1 has been isolated from the network and the update loader cannot find Master 1, so it looks for the master with the lowest node number it can see (in this case it is Inferior Master 2 on UAS1) and connects to that.

The Master 1 queues all updates for UAS1 until such time as it comes back on line. When UAS1 comes back on line, the smsMergeDaemon will query the infMaster process to see if there are any connections to it. If there are any processes connected to the infMaster, the smsMergeDaemon sends a start merge message to the smsMaster. The smsMaster will then update the rest of the network with the information received from UAS1.
If the smsMergeDaemon is not running, the startMerge process may be used instead. startMerge will copy the data from UAS1 to the smsMaster. The smsMaster will then update the rest of the network with the information received from UAS1.

Isolated USMS

The following diagram depicts an isolated USMS.

Where the Master is isolated from the network, each update loader will look for the inferior master with the lowest node number and will connect to that.

Continued on next page
Replication in an Unclustered Installation, Continued

Isolated USMS (continued)

In the above case the Master 1 on the USMS has been isolated. The update loader on each node will look for the inferior master with the lowest node number it can find, in this case the update loaders on both UAS1 and UAS2 will find and connect to inferior master 2 on UAS1. When the USMS comes back into the network, the smsMergeDaemon will check will each UAS infMaster process to see if there are any connections to them. In this case, there are connections to the UAS1 infMaster process (node 2) from the UAS2 (node 3). So the smsMergeDaemon will run startMerge against UAS1. startMerge will copy UAS1’s data across to the USMS. The smsMaster will then attempt to update both UASs with the new data from UAS1.

All nodes isolated The following diagram depicts all nodes isolated.
Replication in an Unclustered Installation, Continued

All nodes isolated (continued)

Where all nodes in the network are isolated, they will each connect to the inferior master with the lowest node number that they can see. In the above example this will result in the update loader on UAS1 connecting to inferior master 2 on UAS1 (node 2), and the update loader on UAS2 connecting to the inferior master 3 (node 3).

As the UASs reconnect to the USMS and reestablish a reliable heartbeat, the smsMergeDaemons run startMerge against each UAS to copy the data across to the USMS. Then the SMS will replicate the data out to the available UASs.

Merging nodes

If a infMaster is acting as a Superior Master, it will collect update requests in a table on the local UAS. When the smsMaster (or another infMaster with a higher node number) reconnects, all local update requests must be forwarded to the new Superior Master node and replicated.

The process for completing this task is known as a merge. Usually, the smsMergeDaemon will initiate a merge automatically when the connection has stabilised. However, it is also possible to start a merge by hand by invoking the startMerge process from the command line.

For more information about using startMerge, see startMerge (on page 187).

Description of resync processes and executables

This table describes the roles of the components involved in the resync process.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsMaster</td>
<td>The smsMaster collects update requests in the pending update queue until the destination updateLoader acknowledges a successful update.</td>
<td>smsMaster (on page 110)</td>
</tr>
<tr>
<td></td>
<td>If the smsMaster cannot connect to a updateLoader, it will collect pending updates until a new connection to the updateLoader is made.</td>
<td></td>
</tr>
<tr>
<td>resyncServer</td>
<td>Takes a snapshot of the SMF and sends it to the compareResyncReceive process on the UAS. One resyncServer will be started for each resync commenced.</td>
<td>resyncServer</td>
</tr>
<tr>
<td>smsCompareResyncServer</td>
<td>Reads configuration information from the configuration file created by resyncServer and starts a resync.</td>
<td>smsCompareResyncServer (on page 162)</td>
</tr>
</tbody>
</table>

Continued on next page
### Description of resync processes and executables (continued)

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>compareResyncReceiver</td>
<td>Updates the UAS with the data from the SMF (sent by resyncServer on the USMS).</td>
<td></td>
</tr>
<tr>
<td>smsCompareResyncClient</td>
<td>Receives information from smsCompareResyncServer and updates the UAS.</td>
<td><em>smsCompareResyncClient</em> (on page 159)</td>
</tr>
<tr>
<td>updateLoader</td>
<td>When a resync is started, the updateLoader will stop making updates to the UAS. Instead it writes the updates to the following file: <code>&lt;nodenum&gt;-queuedOrders.dat</code> When the resync is completed, the queued update orders are processed as normal.</td>
<td><em>updateLoader</em> (on page 150)</td>
</tr>
</tbody>
</table>
replication.def File

replication.def defines default values for all the replication executables on the node it is on. Any of the defaults may be overridden on the command line when the executable is started.

Example: MAX PENDING= 200 can be overridden when starting an smsMaster by adding the command line parameters -maxpending 400 (no spaces in the parameter and all lower case).

Note: Ensure that the heartbeat settings for both ends of a heartbeat are set to the same value. Otherwise the connection will be repeatedly dropped.

This file is located in the following directory:
/IN/service_packages/SMS/etc/

Parameters - replication.def

The replication accepts the following configurable parameters. If they are not defined in the file, the default value will be used.

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COMMIT IDLE TIME</td>
<td>100</td>
<td>Timeout period (in msecs) for the Update Receiver (Update Loader) to become idle after an Update Request (Update Order) and commit.</td>
</tr>
<tr>
<td>COMMIT BUSY TIME</td>
<td>10000</td>
<td>Timeout period (in msecs) for the Update Receiver or updateLoader to commit a change even if it remains continuously busy.</td>
</tr>
<tr>
<td>CONFIG DIR</td>
<td>/IN/service_packages /SMS/etc</td>
<td>The directory where the replication. config file is stored. This parameter has been included for future development, it is recommended that the default is always used.</td>
</tr>
<tr>
<td>CONN RETRY TIME</td>
<td>0</td>
<td>Time (in secs) before an updateLoader tries to reconnect to a Master Replicator if none is available. If set to 0, no re-attempt is made.</td>
</tr>
<tr>
<td>CONNECTION TIMEOUT</td>
<td>1</td>
<td>Timeout (in secs) before an attempted connection to a Master is terminated and alternative is tried.</td>
</tr>
<tr>
<td>HB PERIOD</td>
<td>10</td>
<td>Heartbeat period (in secs), should be consistent across all platforms. Not advisable to take below 3 secs.</td>
</tr>
<tr>
<td>HB TIMEOUT</td>
<td>10</td>
<td>Heartbeat timeout period (in secs) used by smsMergeDaemon, before a heartbeat is considered late (generally set to the same as HB PERIOD).</td>
</tr>
<tr>
<td>HB TOLERANCE</td>
<td>250</td>
<td>Heartbeat tolerance time (in msecs). Default is generally used.</td>
</tr>
<tr>
<td>HTML DIR</td>
<td>/IN/html</td>
<td>The directory where html files are written.</td>
</tr>
</tbody>
</table>

Continued on next page
### Parameters - replication.def (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONG TIMEOUT</td>
<td>60</td>
<td>Heartbeat timeout period (in secs) used by smsMergeDaemon to check if the connections to smsMaster and the node to be merged are stable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If they have both been responding to heartbeats within the time specified in &quot;LONG TIMEOUT&quot;, the merge takes place.</td>
</tr>
<tr>
<td>MASTER PORT</td>
<td>12343</td>
<td>The TCP port that Master Replicators listen for connections on. Generally the default is used.</td>
</tr>
<tr>
<td>MAXMASTERS NODES</td>
<td>8</td>
<td>Defines the number of master nodes used.</td>
</tr>
<tr>
<td><strong>MAX_CONNNECTION_TIME</strong></td>
<td>100000000</td>
<td>Used by smsMergeDaemon.</td>
</tr>
<tr>
<td>MAX_PENDING</td>
<td>10000</td>
<td>Used by Master Replicators to determine maximum size of their pending updates queue (the max number of outstanding updates that are stored before an unconnected updateLoader is considered &quot;Out Of History&quot;).</td>
</tr>
<tr>
<td>MAX.RoundT RIP</td>
<td>3</td>
<td>Used by smsMergeDaemon.</td>
</tr>
<tr>
<td>MERGE_INTERVAL</td>
<td>600</td>
<td>Used by smsMergeDaemon.</td>
</tr>
<tr>
<td>NODE ID</td>
<td>274</td>
<td>Used by an updateLoader to define its replication node number (must be overridden on command line if more than one updateLoader is running on the same UAS machine).</td>
</tr>
<tr>
<td>ORACLE USER</td>
<td>Oracle</td>
<td>Oracle username and associated password normally of the form user/password. Operator accounts are used to maintain security. It is recommended that this is left as &quot;ORACLE USER=/&quot;.</td>
</tr>
<tr>
<td>POLLING INTERVAL</td>
<td>50000</td>
<td>Used to specify the polling interval (in usecs) when the smsMaster is not receiving replication updates.</td>
</tr>
<tr>
<td>QUEUE Warn THRESH</td>
<td>50</td>
<td>The threshold intervals at which warnings are sent to the error log to indicate an increasing/decreasing pending updates queue.</td>
</tr>
</tbody>
</table>

*Continued on next page*
### Parameters - replication.def (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>QUEUE ERR THRESH</td>
<td>200</td>
<td>The threshold intervals at which a warning is turned into an error and sent to the error log to indicate an increasing/decreasing pending updates queue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> To work correctly, this must be greater than the QUEUE WARN THRESH value.</td>
</tr>
<tr>
<td>QUEUE CRIT THRESH</td>
<td>400</td>
<td>The threshold intervals at which a warning or error is turned into a critical error and sent to the error log to indicate an increasing/decreasing pending updates queue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> To work correctly, this must be greater than the QUEUE ERR THRESH value.</td>
</tr>
<tr>
<td>REP_PATH</td>
<td>/IN/service_packages/SMS/etc/replication.config</td>
<td>Used by smsMergeDaemon.</td>
</tr>
<tr>
<td>REPORT DIR</td>
<td>/IN/service_packages/SMS/output/Replication</td>
<td>The directory where replication reports (e.g. merge reports and database comparison reports) are stored.</td>
</tr>
<tr>
<td>RESYNC DIR</td>
<td>IN/service_packages/SMS/tmp</td>
<td>The directory where an updateLoader's pendingUpdates.dat file is stored during a resync.</td>
</tr>
<tr>
<td>SECONDARY DELAY</td>
<td>100000</td>
<td>Initial time (in usecs) that the primary network has to establish a connection before attempting to connect over the secondary network as well. A value of 0 means both networks are attempted immediately.</td>
</tr>
<tr>
<td>SMS_PORT</td>
<td>7</td>
<td>Used by smsMergeDaemon.</td>
</tr>
<tr>
<td>STDERRON</td>
<td>0</td>
<td>Set to 1 to ensure errors go to stdout as well as in the error log.</td>
</tr>
<tr>
<td>STATSKEY</td>
<td>270198</td>
<td>Shared memory key for updateLoader replication statistics. It is recommended that this always use the default, if the default is not used part of the statistics gathering system (dm_sys) will no longer be able to find the statistics.</td>
</tr>
<tr>
<td>TICK_TIME</td>
<td>1000</td>
<td>Used by smsMergeDaemon.</td>
</tr>
</tbody>
</table>

#### Example replication.def file

Here is an example replication.def file for an USMS machine:

```
MAX PENDING=5000
HB PERIOD=5
HB TIMEOUT=5
```

Continued on next page
Example replication.def file (continued)

```
CONNECTION TIMEOUT=2
ORACLE USER=/
CONFIG DIR=/IN/service_packages/SMS/etc
REPORT DIR=/IN/service_packages/SMS/output/Replication
HTML DIR=/IN/html
SECONDARY DELAY=200000
COMMIT BUSY TIME=1000
POLLING INTERVAL=50000
```
### Introduction

The replication.config file is a binary configuration file that defines the current specific replication set-up. It is a binary representation of the replication set-up within the SMF created by the repConfigWrite process.

This file is used by all replication nodes on a machine, and must be:

- the same on each machine, and
- accessible by each node.

The file is written to the directory specified by the output parameter.

### Generating replication.config

This file is usually created by clicking **Create Config File** on the Table Replication tab of the Node Management Screens.

### Example replication.config

This text shows an example of a replication.config file which has been converted using smsDumpRepConfig.

```plaintext
smsDumpRepConfig: File
/IN/service_packages/SMS/etc/replication.config
smsDumpRepConfig: (PAD = 0)
smsDumpRepConfig: Short listing. Use -v (verbose) for full listing

TABLE [ACS_CALL_PLAN]
TABLE [ACS_CALL_PLAN_PROFILE]
TABLE [ACS_CALL_PLAN_STRUCTURE]
TABLE [ACS_CLI_CALL_PLAN_ACTIVATION]
TABLE [ACS_CUSTOMER]
TABLE [ACS_CUSTOMER_CLI]
TABLE [ACS_CUSTOMER_SN]
TABLE [ACS_FN_TYPE]
TABLE [ACS_GLOBAL_PROFILE]
TABLE [ACS_LANGUAGE]
TABLE [ACS_NETWORK_KEY]
TABLE [ACS_SN_CALL_PLAN_ACTIVATION]
TABLE [SMF_ALARM_MESSAGE]
TABLE [SMF_STATISTICS]
TABLE [SMF_STATISTICS_DEFN]

smsDumpRepConfig: Replication Groups configured for each node...

----------

NODE NUMBER [1] Prim (192.168.0.173) Sec (0.0.0.0)
GROUP [ACS_CUSTOMER] [Prim=-1] Min=('+0','','') Max=('9','','')
GROUP [ACS_FN_TYPE] [Prim=-1] Min=('+0','','') Max=('9','','')
GROUP [ACS_CALL_PLAN_PROFILE] [Prim=-1] Min=('+0','','')
GROUP [ACS_CALL_PLAN_STRUCTURE] [Prim=-1] Min=('+0','','')
GROUP [ACS_CALL_PLAN] [Prim=-1] Min=('+0','','')
GROUP [ACS_CUSTOMER_CLI] [Prim=-1] Min=('+0','','')
GROUP [ACS_CUSTOMER_SN] [Prim=-1] Min=('+0','','')

----------

Continued on next page
Example replication.config (continued)

GROUP [ACS_LANGUAGE] [Prim=-1] Min=('0','','') Max=('9','','')
GROUP [SMF_STATISTICS_DEFN] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_CLI_CALL_PLAN_ACTIVATION] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_GLOBAL_PROFILE] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_NETWORK_KEY] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_SN_CALL_PLAN_ACTIVATION] [Prim=-1] Min=('0','','')
Max=('9','','')

GROUP [ACS_CUSTOMER] [Prim=-1] Min=('0','','') Max=('9','','')
GROUP [ACS_FN_TYPE] [Prim=-1] Min=('0','','') Max=('9','','')
GROUP [ACS_CALL_PLAN_PROFILE] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_CALL_PLAN_STRUCTURE] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_CALL_PLAN] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_CUSTOMER_CLI] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_LANGUAGE] [Prim=-1] Min=('0','','') Max=('9','','')
GROUP [SMF_STATISTICS_DEFN] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_CLI_CALL_PLAN_ACTIVATION] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_GLOBAL_PROFILE] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_NETWORK_KEY] [Prim=-1] Min=('0','','')
Max=('9','','')
GROUP [ACS_SN_CALL_PLAN_ACTIVATION] [Prim=-1] Min=('0','','')
Max=('9','','')

For more information, see:

- replication.config File (on page 39)
- smsDumpRepConfig (on page 175), and
- SMS User's Guide.
Chapter 3

Replication Check

Overview

<table>
<thead>
<tr>
<th>Introduction</th>
<th>This chapter explains replication check and data resynchronisation processes used in SMS</th>
</tr>
</thead>
<tbody>
<tr>
<td>In this chapter</td>
<td>This chapter contains the following topics.</td>
</tr>
<tr>
<td>Replication Checks</td>
<td>42</td>
</tr>
<tr>
<td>Database Comparisons</td>
<td>44</td>
</tr>
<tr>
<td>Database Resynchronisations</td>
<td>46</td>
</tr>
<tr>
<td>Auditing</td>
<td>48</td>
</tr>
</tbody>
</table>
Replication Checks

**Description**
SMS provides a replication check mechanism to enable operators to check the replication of data across their services network.

A replication check will perform a comparison of the SMF data on each replication node. Once the comparison is complete a report will be generated detailing any discrepancies. No data is changed.

Depending on the size of the data set there may be sizable performance impact on the client node and care should be taken to perform such a check outside of peak times.

**Replication check components**
This table describes the components involved in replication check process.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMF</td>
<td>The main database on the USMS.</td>
<td></td>
</tr>
<tr>
<td>UAS</td>
<td>The databases on the UASs. They hold a subset of the data on the SMF.</td>
<td></td>
</tr>
<tr>
<td>writeHTMLDirFile</td>
<td>Updates index.html to include new replication, comparison and resynchronisation reports.</td>
<td></td>
</tr>
</tbody>
</table>

Continued on next page
Replication Checks, Continued

Replication check components (continued)

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsCompareResyncServer</td>
<td>Performs database resynchronisations and comparisons on the superior node.</td>
<td>smsCompareResyncServer (on page 162)</td>
</tr>
<tr>
<td>smsCompareResyncClient</td>
<td>Performs database resynchronisations and comparisons on the inferior node.</td>
<td>smsCompareResyncClient (on page 159)</td>
</tr>
<tr>
<td>inetCompareServer</td>
<td>Accepts Replication Check requests from the Replication Check screen.</td>
<td>inetCompareServer</td>
</tr>
</tbody>
</table>

Replication check process

The replication check process follows these stages.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The <code>run all</code> command starts <code>inetCompareServer</code> as configured in the replication check report.</td>
</tr>
<tr>
<td>2</td>
<td><code>inetCompareServer</code> configures and starts <code>smsCompareResyncServer</code> (on page 162) and <code>writeHTMLDirFile</code>.</td>
</tr>
<tr>
<td>3</td>
<td><code>smsCompareResyncClient</code> (on page 159) handles the other end of the replication check.</td>
</tr>
<tr>
<td>4</td>
<td><code>writeHTMLDirFile</code> updates <code>index.html</code> to include the new report for display in the screens.</td>
</tr>
</tbody>
</table>
compareNode is used to initiate a full database comparison of an UAS database with the definitive copy in SMF db. This ensures that an UAS's data is consistent with the SMF database. Under normal conditions, this should always be the case, but there may be a time (for example, after multiple failures) where the System Administrator wants to check an UAS is consistent.

compareNode tool requests a comparison between the contents of SMF and one other node, by invoking comparisonServer. Comparisons are a more time-efficient method than resyncs. comparisonServer compares all the entries of all tables which are defined to be replicated to specified updateLoader node.

Here is a diagram that shows the elements involved in a database comparison process.
Database Comparisons, Continued

This table describes the components involved in database comparison process.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>compareNode</td>
<td>Initiates a full database comparison of an UAS with the definitive copy in SMF. Starts comparisonServer.</td>
<td>compareNode (on page 153)</td>
</tr>
<tr>
<td>inputBootstrap</td>
<td>Creates configuration files for smsCompareResyncServer from replication.config.</td>
<td>inputBootstrap (on page 155)</td>
</tr>
<tr>
<td>smsMaster</td>
<td>Receives update requests and forwards them to the SMF.</td>
<td>smsMaster (on page 110)</td>
</tr>
<tr>
<td>comparisonServer</td>
<td>Creates configuration files for smsCompareResyncServer from replication.config.</td>
<td></td>
</tr>
<tr>
<td>smsCompareResyncClient</td>
<td>Performs database resynchronisations and comparisons on the inferior node.</td>
<td>smsCompareResyncClient (on page 159)</td>
</tr>
<tr>
<td>writeHTMLDirFile</td>
<td>Updates index.html to include new replication, comparison and resynchronisation reports.</td>
<td></td>
</tr>
</tbody>
</table>

The database comparison process follows these stages.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>compareNode sends a comparison request to smsMaster.</td>
</tr>
<tr>
<td>2</td>
<td>smsMaster configures and starts comparisonServer.</td>
</tr>
<tr>
<td>3</td>
<td>inputBootstrap provides configuration for comparisonServer from replication.config.</td>
</tr>
<tr>
<td>4</td>
<td>comparisonServer configures and starts smsCompareResyncServer and writeHTMLDirFile.</td>
</tr>
<tr>
<td>5</td>
<td>smsCompareResyncClient handles the other end of the comparison.</td>
</tr>
<tr>
<td>6</td>
<td>writeHTMLDirFile updates index.html to include the new report for display in the screens.</td>
</tr>
</tbody>
</table>
Database Resynchronisations

Description

It is possible for nodes to become out of sync to the point where normal recovery processes will not rectify the problem. This may happen if there is a network failure for a long period of time, or if there is a fault in the replication process.

If the databases are out of sync, a resynchronisation must be run. Resyncs compare the data in two specified nodes and updates the inferior database with the different information in the superior database.

Resyncs will run automatically if:

- the updateLoader is started with the -resync flag, and there is a queuedOrders.dat file
- the smsMaster has marked that updateLoader node as out of history
- the smsMaster is connected to by an updateLoader which is not in its pending updates queue, or
- a resync instruction is included in a smsTaskAgent file.

Resyncs can also be run from the command line. For more information about running resynchronisations, see resync (see "resyncServer" on page 158).

Here is a diagram that shows the elements involved in a database resynchronisation process.

---

*Continued on next page*
Database Resynchronisations, Continued

This table describes the components involved in database resynchronisation process.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>inputBootstrap</td>
<td>Creates configuration files for smsCompareResyncServer from replication.config.</td>
<td>inputBootstrap (on page 155)</td>
</tr>
<tr>
<td>resyncServer</td>
<td>Starts smsCompareResyncServer and inputBootstrap for resyncs.</td>
<td>resyncServer (on page 158)</td>
</tr>
<tr>
<td>smsCompareResyncClient</td>
<td>Performs database resynchronisations and comparisons on the inferior node.</td>
<td>smsCompareResyncClient (on page 159)</td>
</tr>
<tr>
<td>writeHTMLDirFile</td>
<td>Updates index.html to include new replication, comparison and resynchronisation reports.</td>
<td></td>
</tr>
</tbody>
</table>

The resynchronisation process follows these stages.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>If the resync has been started from the command line using resync, resync will send a resynchronisation request to smsMaster.</td>
</tr>
<tr>
<td>2</td>
<td>smsMaster will send a resynchronisation request to resyncServer.</td>
</tr>
<tr>
<td>3</td>
<td>resyncServer sends a request to inputBootstrap.</td>
</tr>
<tr>
<td>4</td>
<td>inputBootstrap reads data from replication.config and creates a configuration file for smsCompareResyncServer.</td>
</tr>
<tr>
<td>5</td>
<td>resyncServer starts smsCompareResyncServer in resync mode.</td>
</tr>
<tr>
<td>6</td>
<td>smsCompareResyncServer connects to the SMF and smsCompareResyncClient on the inferior node.</td>
</tr>
<tr>
<td>7</td>
<td>smsCompareResyncClient accepts the connection and connects to the UAS.</td>
</tr>
<tr>
<td>8</td>
<td>smsCompareResyncServer and smsCompareResyncClient resync the databases.</td>
</tr>
<tr>
<td>9</td>
<td>smsCompareResyncServer writes a resynchronisation report to /IN/html/output/SMS/resync/&lt;inferior node number&gt;/yyyymmddhhmss.report</td>
</tr>
<tr>
<td>10</td>
<td>writeHTMLDirFile updates /IN/html/output/SMS/resync/&lt;inferior node number&gt;/index.html to include the new report.</td>
</tr>
</tbody>
</table>
Auditing

SMS provides an auditing function for all services implemented through it. It tracks all changes made to SMF database and stores them in SMF_AUDIT table. It records:

- User’s userid
- IP address of terminal the change was made from
- timestamp of change
- table changed, and
- copy of record before change and a copy of record after change.

Most database tables also have Change User and Change Date columns which record:

- username of user which last changed the contents of a table, and
- date this change was made at.

Audit information is produced as a report by running listAudit.sh.

Run either as a cron job or from command line using this command:

```
listAudit.sh <usr/pwd> [<start date>] [<end date>] [<db user>] [<table>]
```
Overview

Introduction
This chapter introduces the SMS Configuration Management editor and explains its background processes.

In this chapter
This chapter contains the following topics.
- Configuration Management Editor .................................................. 50
- SMS - Xerlin Integration ................................................................. 51
- Screen and Access Permissions ...................................................... 52
- Distributing Configuration Files .................................................... 53
Configuration Management Editor

Introduction

The Configuration Management editor is a Xerlin based XML editor. Configuration data is displayed in the hierarchical form of a collapsible tree as in the eserv.config files. The editor is integrated into the SMS application and can be invoked from the standard SMS management screens.

For information on how to access and use the Configuration Management editor, see SMS User's Guide, chapter Configuration Management.

Schema Definition File

Xerlin supports the schema definition document (XSD) to define the exact format and data structures of the XML permitted. Documents are only considered valid if they satisfy the requirements of the schema with which they have been associated.

The schema file used for validating the configuration data for all NCC applications, which support the use of this editor, is esgConfig.xsd.

Editing XML Documents

The Xerlin editor typically edits the the working copy eserv.config file, esgConfigWorking.xml.

All changes made using the editor are saved to the working copy.

During deployment, the last saved changes on the working document are copied to the master configuration file, esgConfigMaster.xml. The master XML configuration is then converted to an eserv.config format and sent out to the relevant platforms for synchronisation.

For information on editing configurations using the Configuration Management editor, see SMS User's Guide, chapter Using Configuration Management.
SMS - Xerlin Integration

**High Level Architecture**

The key elements of the integrated design are:

- The Xerlin based Configuration Management editor is included as part of the SMS package.
- The editor is launched from within the SMS main menu in the same browser session as currently used by SMS.
- A configuration distribution mechanism facilitates distribution of data to eserv.config files saved on other NCC Universal Service Platforms (USPs).
- NCC applications which support this mechanism are managed via the master configuration document.
- Permissions to access the configuration editor are enabled following the installation of the first application which supports this feature, example the Diameter Control Agent (DCA).

**Xerlin License Requirements**

The Xerlin XML Editor is covered by the Open Source License.

For more information on the terms and rights of usage, read the Xerlin license document available at: http://www.xerlin.org/LICENSE.txt

(http://www.xerlin.org/LICENSE.txt)
Screen and Access Permissions

The Configuration Management editor can be accessed using the Operator Functions menu from the SMS Main screen.

Granting permissions to Xerlin for accessing and editing the master configuration documents used by various NCC applications are set by the applications themselves as a one-time process.
Distributing Configuration Files

**Description**

After editing configuration data using the Xerlin editor, the XML files are distributed to the respective platforms which utilise them. The distribution process is initiated when the Configuration -> Deploy option is selected on the Operator Functions menu from the SMS Main screen.

This process takes place on both:

- Source node (for example, the active USMS node)
- Target node (for example, on the UAS)

**Conversion script**

Prior to deploying the configuration data to the relevant platforms, the XML document edited in Xerlin must be converted into a suitable eserv.config format.

The following script is used for this conversion:

```
SMS/bin/cmnConfigXmlConvert.sh
```

The file generated from this process contains only the modified sections and not all sections present in the working eserv.config file used on that particular node.

**Node identification**

The converted (derived) eserv.config file containing the latest changes made using the Xerlin editor is sent to the nodes defined in the SMF database using the node configuration screens.

For more information on node configuration, see *SMS User’s Guide, chapter Setting up SMS for the First Time - Node Management*.

**Process on Source node**

The following steps describe the distribution process on the source node during deployment:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Identifying any previous configurations that failed due to network problems and resending them. This is achieved by clearing out entries in the subdirectories located in the following directory as each file is successfully sent out: /IN/html/Configuration/eserv/pending</td>
</tr>
<tr>
<td>2</td>
<td>Identifying updates to the master XML configuration based on the modifications to the date/time stamp and checksum. The master XML configuration file is located in: /IN/html/Configuration/xml/esgConfigMaster.xml</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The time interval for checking for updates to the master file is configurable with a default value of 10 seconds.</td>
</tr>
<tr>
<td>3</td>
<td>If the master file has been modified, a copy of the file is saved as backup to the archive directory with the date/time stamp: /IN/html/Configuration/xml/archive</td>
</tr>
<tr>
<td>4</td>
<td>The old (as pointed to in the last_config.xml) and new file names are added to the SMF_AUDIT table.</td>
</tr>
<tr>
<td>5</td>
<td>The last_config.xml is updated to point to the backed up file. The last_config.xml is located in: /IN/html/Configuration/xml/</td>
</tr>
</tbody>
</table>

Continued on next page
### Distributing Configuration Files, Continued

#### Process on Source node (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>The master XML file is converted to an eserv.config format using the cmnConfigXmlConvert.sh script.</td>
</tr>
<tr>
<td></td>
<td>This derived eserv.config file is then saved to the following location with the date/time stamp:</td>
</tr>
<tr>
<td></td>
<td>/IN/html/Configuration/eserv/&lt;derived_eserv&gt;</td>
</tr>
<tr>
<td></td>
<td>For more information on the conversion script, see <strong>Conversion script</strong> (on page 53).</td>
</tr>
<tr>
<td>7</td>
<td>A copy of the derived eserv.config file, consisting of only the sections modified in XML, is copied to the following directory on the target platforms using scp:</td>
</tr>
<tr>
<td></td>
<td>/IN/service_packages/Configuration/eserv</td>
</tr>
<tr>
<td>8</td>
<td>If a problem is encountered while transmitting the file, then the file is copied to a subdirectory under the pending directory on the the source, with the IP address of the target host and the date/time stamp:</td>
</tr>
<tr>
<td></td>
<td>/IN/html/Configuration/eserv/pending/&lt;target_host_ip_address&gt;</td>
</tr>
<tr>
<td></td>
<td>This ensures a recovery mechanism to resend the file later if a reboot occurs or if the daemon was killed.</td>
</tr>
</tbody>
</table>

#### Process on Target node

The following steps describe the distribution process on the target node during deployment:

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Checking for updates to the derived eserv.config file based on the modifications to the date/time stamp.</td>
</tr>
<tr>
<td></td>
<td>The derived eserv.config file is located in:</td>
</tr>
<tr>
<td></td>
<td>/IN/service_packages/Configuration/eserv</td>
</tr>
<tr>
<td>2</td>
<td>If the derived file has been modified, a backup copy is created of the currently live eserv.config file located in:</td>
</tr>
<tr>
<td></td>
<td>/IN/service_packages/</td>
</tr>
<tr>
<td></td>
<td>This backup copy is saved to the backup directory:</td>
</tr>
<tr>
<td></td>
<td>/IN/service_packages/Configuration/eserv/live_backup</td>
</tr>
<tr>
<td>3</td>
<td>Identifying the sections in the derived eserv.config and merging them in the live eserv.config file.</td>
</tr>
<tr>
<td></td>
<td>For more information on updating the eserv.config, see <strong>Modifying eserv.conf on target node</strong> (on page 55).</td>
</tr>
<tr>
<td>4</td>
<td>Performing validation checks on the eserv.config. If this fails, the backup copy is reverted back as the live eserv.config and an alarm is raised indicating the problem.</td>
</tr>
<tr>
<td>5</td>
<td>Notifying relevant processes of the change in configuration file through a SIGHUP process.</td>
</tr>
<tr>
<td></td>
<td>For more information on the notification process, see <strong>Reloading configuration data</strong> (on page 55).</td>
</tr>
</tbody>
</table>

*Continued on next page*
Distributing Configuration Files, Continued

Modifying eserv.config

smsConfigSurgeon is used for merging changes from the derived eserv.config file into the live eserv.config file. It accurately identifies the changed section labels in the derived file and merges the matching sections in the live file.

smsConfigSurgeon takes the following parameters:

- Source file
- Target file
- Section name, and
- Action, i.e., merge

The file and group permissions are initially set to the User smf_oper: Oracle and are later altered by the respective packages and patches installed.

Reloading configuration data

After the live eserv.config file is updated on the target node, a notification is sent to the relevant applications and processes using both SIGHUP and SLEE Management event RELOAD_CONFIG as appropriate.

It is important to identify those processes which need to reload their configuration, based on the changed components of the configuration file. This information is stored in the SMF_APPLICATION_CONFIG table in the SMF database. The table contains a replicated table mapping between the SMF_APPLICATION.APP_ID and the relevant configuration sections.
Chapter 5

Configuring the Environment

Overview

Introduction

This chapter explains the steps required to configure SMS.

In this chapter

This chapter contains the following topics.

- Configuration Overview ................................................................. 58
- Configuring the Resource Group in the Clustered Environment .......... 60
- Configuring Replication Files ......................................................... 64
- Configuring the Oracle Listener ..................................................... 66
- Configuring the SNMP Agent ......................................................... 69
- SMF AlarmMessage Format ............................................................ 77
- Defining the Screen Language ....................................................... 79
- Defining the Help Screen Language .............................................. 81
- Setting up the screens ................................................................. 83
- Configuring Nodes ....................................................................... 89
- Installing Additional Applications ............................................... 90
Configuration Overview

This topic provides a high level overview of how the SMS application is configured. Configuration details for individual processes are located with the documentation for that process.

Configuration process overview

This table describes the steps involved in configuring SMS for the first time.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | The environment SMS will run in must be configured correctly. This includes:  
      - if the directory SMS was installed into was not the recommended directory, setting the root directory  
      - if this was a clustered installation, configuring the Resource Groups  
      - configuring the Oracle Listener  
      - configuring the SNMP Agent  
      - configuring the location of the EDR directories  
      - configuring the smf_oper profile, and  
      - configuring the webserver. |
| 2     | The replication groups must be configured. |
| 3     | If the default language for the SMS Java Administration screens need changing, the new default language must be configured. |
| 4     | If the default language for the help system for the SMS Java Administration screens need changing, the new default language must be configured. |
| 5     | The SMS screen-based configuration must be completed. This includes checking node configuration and statistics configuration. |

SMS is configured by the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Locations</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
</table>
| SMS Java Administration screens | USMS | The SMS screens provide a graphical interface for configuring many parts of SMS including:  
   - replication  
   - statistics  
   - alarm filtering, and  
   - reports. | SMS User’s Guide |
| replication.def | all machines with running replication agents | This file specifies the configuration parameters for replication. These parameters may also be specified on the command-line for each application. | replication.def File (on page 35) |

Continued on next page
## Configuration Overview, Continued

### Configuration components (continued)

<table>
<thead>
<tr>
<th>Component</th>
<th>Locations</th>
<th>Description</th>
<th>Further Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>replication.config</td>
<td>all machines with running replication agents</td>
<td>This file holds a binary version of the configuration held in the SMF. It is copied out to all machines and is required by all replication agent.</td>
<td>replication.config File (on page 39)</td>
</tr>
<tr>
<td>logjob.conf</td>
<td>all USMSs</td>
<td>This file is automatically generated when the smsSms package is installed.</td>
<td>logjob.conf (on page 108)</td>
</tr>
<tr>
<td>snmp.cfg</td>
<td>all USMSs</td>
<td>This file configures the SNMP agent's details.</td>
<td>Configuring the SNMP Agent (on page 69)</td>
</tr>
<tr>
<td>repsib.cfg</td>
<td>all UASs</td>
<td>This is the template file for the updateRequester program. Other applications will typically append template definitions to this file when they are installed, and remove them when they are uninstalled.</td>
<td></td>
</tr>
</tbody>
</table>
Configuring the Resource Group in the Clustered Environment

Overview

Certain tasks performed by the cluster will only require one instance running across all cluster nodes, for example an application which modifies a shared data source. There must be a mechanism in place to monitor the running processes and make sure they are restarted should any problems arise. This is similar to normal UNIX initab functionality with the caveat that a process can be restarted on any of the cluster nodes (failover).

Configuration of resource groups must be completed on each node in the cluster.

Starting the webservice failover

Follow these steps to start the httpd failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the ESERVHttpd directory.  
      | Example command: cd /opt/ESERVHttpd |
| 2    | Read the readme file. |
| 3    | Stop apache.  
      | Example command: /usr/apache/bin/apachectl stop |
| 4    | Change to the util directory within the httpd directory.  
      | Example command: cd util/ |
| 5    | Start httpd failover using the following command.  
      | startHttpd -h <hostname> -p "<port number>/tcp"  
      | Where:  
      | • hostname is the shared hostname for the USMS cluster  
      | • port is the port number the webserver will accept httpd requests on  
      | Example command: startHttpd -h smpVirtualCluster -p "80/tcp" |

For more information about shared hostnames for clustered machines, see the Oracle documentation.

Starting the sshd failover

Follow these steps to start the sshd failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the ESERVSShd directory.  
      | Example command: cd /opt/ESERVSShd |
| 2    | Read the readme file. |
| 3    | Stop the sshd.  
      | Example command: /etc/init.d/sshd stop |
| 4    | Change to the util directory in ESERVSShd.  
      | Example command: cd util |

Continued on next page
### Configuring the Resource Group in the Clustered Environment, Continued

**Starting the sshd failover (continued)**

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 5    | Start the sshd failover with the following command:  
      \[ \text{startSshd -h <hostname> -p "<port>/tcp"} \]  
      Where:  
      - hostname is the shared hostname of the USMS cluster  
      - port is the port number the sshd should be running on  
      Example command:  
      \[ \text{startSshd -h smpVirtualCluster -p "22/tcp"} \]  
      For more information about shared hostnames for clustered machines, see the Oracle documentation. |

**Starting the smsAlarmDaemon failover**

Follow these steps to start the \textit{smsAlarmDaemon} (on page 94) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Unset the \$HOSTNAME environmental variable.  
      \textbf{Example commands:}  
      \begin{itemize}  
        \item \texttt{echo \$HOSTNAME}  
        \item \texttt{unset \$HOSTNAME}  
        \item \texttt{echo \$HOSTNAME}  
      \end{itemize} |
| 2    | Change to the OracleSmsAlarmDaemon/util directory.  
      \textbf{Example command:}  
      \[ \text{cd /opt/OracleSmsAlarmDaemon/util} \] |
| 3    | Start the smsAlarmDaemon.  
      \textbf{Example command:}  
      \[ \text{./startSmsAlarmDaemon} \]  
      \textbf{Result:} The following information will be sent to stdout:  
      \begin{itemize}  
        \item Creating a scalable instance...  
        \item Registering resource type \texttt{<Oracle.SmsAlarmDaemon>...done.}  
        \item Creating scalable resource group \texttt{<SmsAlarmDaemon-sarg>...done.}  
        \item Creating resource \texttt{<SmsAlarmDaemon-sars> for the resource type <Oracle.SmsAlarmDaemon>...done.}  
        \item Bringing resource group \texttt{<SmsAlarmDaemon-sarg> online...done.}  
      \end{itemize} |

**Starting the smsAlarmRelay failover**

Follow these steps to start the \textit{smsAlarmRelay} (on page 99) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the OracleSmsAlarmRelay directory.  
      \textbf{Example command:}  
      \[ \text{cd /opt/OracleSmsAlarmRelay} \] |
| 2    | Read the readme file. |
| 3    | Change to the util directory in the OracleSmsAlarmRelay.  
      \textbf{Example command:}  
      \[ \text{cd util} \]  
      \textit{Continued on next page}
### Configuring the Resource Group in the Clustered Environment, Continued

#### Starting the smsAlarmRelay failover (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 4    | Start the smsAlarmRelay.  
**Example command:** ./startSmsAlarmRelay  
**Result:** The following information will be sent to stdout:  
Creating a failover instance...  
Registering resource type <Oracle.SmsAlarmRelay>...done.  
Creating failover resource group <SmsAlarmRelay-harg>...done.  
Creating resource <SmsAlarmRelay-hars> for the resource type <Oracle.SmsAlarmRelay>...done.  
Bringing resource group <SmsAlarmRelay-harg> online...done. |

#### Starting the smsNamingServer failover

Follow these steps to start the *smsNamingServer* (on page 111) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the OracleSmsNamingServer/util directory.  
**Example command:** cd /opt/OracleSmsNamingServer/util  
Start the smsNamingServer failover.  
**Example command:** ./startSmsNamingServer  
**Result:** The following information is sent to stdout:  
Creating a scalable instance...  
Registering resource type <Oracle.SmsNamingServer>...done.  
Creating scalable resource group <SmsNamingServer-sarg>...done.  
Creating resource <SmsNamingServer-sars> for the resource type <Oracle.SmsNamingServer>...done.  
Bringing resource group <SmsNamingServer-sarg> online...done. |

#### Starting the smsReportScheduler failover

Follow these steps to start the *smsReportScheduler* (on page 114) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the OracleSmsReportScheduler/util directory.  
**Example command:** cd /opt/OracleSmsReportScheduler/util  
Start the smsReportScheduler failover.  
**Example command:** ./startSmsReportScheduler  
**Result:** The following information is sent to stdout:  
Creating a failover instance...  
Registering resource type <Oracle.SmsReportScheduler>...done.  
Creating failover resource group <SmsReportScheduler-harg>...done.  
Creating resource <SmsReportScheduler-hars> for the resource type <Oracle.SmsReportScheduler>...done.  
Bringing resource group <SmsReportScheduler-harg> online...done. |

*Continued on next page*
### Configuring the Resource Group in the Clustered Environment, Continued

#### Starting the smsReportsDaemon on failover

Follow these steps to start the `smsReportsDaemon` (on page 112) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the OracleSmsReportsDaemon/util directory.  
**Example command:** `cd /opt/OracleSmsReportsDaemon/util` |
| 2    | Start the smsReportsDaemon failover.  
**Example command:** `./startSmsReportsDaemon` |

**Result:** The following information is sent to stdout:
- Creating a scalable instance ...
- Registering resource type `<Oracle.SmsReportsDaemon>`...done.
- Creating scalable resource group `<SmsReportsDaemon-sarg>`...done.
- Creating resource `<SmsReportsDaemon-sars>` for the resource type `<Oracle.SmsReportsDaemon>`...done.
- Bringing resource group `<SmsReportsDaemon-sarg>` online...done.

#### Starting the smsStatsThreshold failover

Follow these steps to start the `smsStatsThreshold` (on page 118) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the OracleSmsStatsThreshold/util directory.  
**Example command:** `cd /opt/OracleSmsStatsThreshold/util` |
| 2    | Start the smsStatsThreshold failover.  
**Example command:** `./startSmsStatsThreshold` |

**Result:** The following information is sent to stdout:
- Creating a failover instance ...
- Registering resource type `<Oracle.SmsStatsThreshold>`...done.
- Creating failover resource group `<SmsStatsThreshold-harg>`...done.
- Creating resource `<SmsStatsThreshold-hars>` for the resource type `<Oracle.SmsStatsThreshold>`...done.
- Bringing resource group `<SmsStatsThreshold-harg>` online...done.

#### Starting the smsTaskAgent failover

Follow these steps to start the `smsTaskAgent` (on page 120) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the OracleSmsTaskAgent/util directory.  
**Example command:** `cd /opt/OracleSmsTaskAgent/util` |
| 2    | Start the smsTaskAgent.  
**Example command:** `./startSmsTaskAgent` |

**Result:** The following information will be sent to stdout:
- Creating a scalable instance ...
- Registering resource type `<Oracle.SmsTaskAgent>`...done.
- Creating scalable resource group `<SmsTaskAgent-sarg>`...done.
- Creating resource `<SmsTaskAgent-sars>` for the resource type `<Oracle.SmsTaskAgent>`...done.
- Bringing resource group `<SmsTaskAgent-sarg>` online...done.
Configuring Replication Files

Introduction

There are two configuration files for replication that may be changed by the administrator.

- replication.def
- replication.config

The replication.config file

The replication.config is created and changed through the Node Management screens in the SMS screens. The user must move tables on the screen from the Available Replication Groups list to the node they are to be replicated to in the Allocated Replication Groups list. Clicking Create Config File will produce a new replication.config file.

The previous configuration will be deleted prior to the new configuration being loaded. This will not necessitate the application being restarted but will cause disruption to service on any of the UASs.

The replication.config file contains the configuration for the whole network. This will include all configuration details needed for smsMasters and infMasters (if necessary).

Implementing changes to the replication.config file

The new replication.config files take effect after the program called changeConfig is run.

If you make the new configuration from the screens this will be immediately. If you make the config from the command line the change can be scheduled.

The replication.def file

The replication.def is configured when the application is installed and should not need to be updated. It contains parameters that may be changed by the operator on start-up.

Implementing change to replication.def

Since replication.def is read only when the application starts up, if it does need to be updated and changes are made, the application (updateLoader, infMaster or smsMaster) must be restarted for these changes to take effect. Once restarted, these changes will take effect immediately.

Replication.def is held on each node in the same directory as the application (updateLoader, infMaster or smsMaster). If changes are made to the UAS configuration the infMaster and the updateLoader must be restarted.

Where changes are to be made to the SMS configuration, the smsMasters must be restarted. In an unclustered installation, the smsMaster must be shut down by merging it with an infMaster to avoid loss of data and update information.

Example replication.def file

Here is an example replication.def file: (this is the default replication.def that is installed as part of the package installation).

```
# @(#)replication.def   1.2
MAX PENDING=10000
ORACLE USER=/
HB PERIOD=10
HB TIMEOUT=10
HB TOLERANCE=250
CONNECTION TIMEOUT=2
SECONDARY DELAY=100000
CONN RETRY=1
QUEUE WARN THRESH=5
#Some Update Loader values
CONN RETRY TIME=0
```

Continued on next page
Configuring Replication Files, Continued

Example replication.def file (continued)

RESYNC DIR=/IN/service_packages/SMS/tmp
CONFIG DIR=/IN/service_packages/SMS/etc
HTML DIR=/IN/html
REPORT DIR=/IN/service_packages/SMS/output/Replication
Configuring the Oracle Listener

Introduction
Perform this procedure on the USMS machine only; it is not required on any of the UASs.

**Note:** This is not a comprehensive guide to Oracle configuration. Configuring and maintaining a database is a non-trivial task, and if you are unsure how to proceed please consult with your Database Administrator.

In order for the USMS to operate correctly it requires an Oracle listener. The Oracle listener is a process whose job it is to listen for external requests to connect to a database on the machine.

Oracle listener configuration is defined in the file listener.ora.

Creating an Oracle listener file is a DBA task as described in Chapter 5 (Using Sql*Net) of the Understanding Sql*Net document that is shipped with Oracle 7.

This procedure does not intend to describe how to create a listener.ora file, but only how to add support for providing access to Oracle Database instances using the TCP network protocol. In fact, this process of adding TCP support is also described in the documentation, however it is outlined here for quick reference.

Procedure
Follow these steps to configure the Oracle Listener.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Log in as an Oracle user or by becoming the Oracle user with start-up script loading from a root login.  
      **Example command:** `su - oracle`  
      **Note:** This will ensure the path to all the Oracle binaries is correct and that file ownership for Oracle files is preserved. |
| 2    | Change to one of the following directories:  
      `$ORACLE_HOME/network/admin` or  
      `/var/opt/oracle/`  
      **Note:** This file is located in different places depending on the version of Oracle installed and the options selected at the time Oracle was installed. |
| 3    | Edit the `listener.ora` file.  
      **Example command:** `vi listener.ora` |
| 4    | Add an ADDRESS entry to the ADDRESS_LIST.  
      Where:  
      • `KEY=SMF`  
      • `HOST=<Hostname>`  
      • `PORT=1521` |

*Continued on next page*
Configuring the Oracle Listener, Continued

**Procedure** (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>The example below illustrates the configuration for an USMS machine called &quot;marco&quot;:</td>
</tr>
<tr>
<td></td>
<td>LISTENER=(ADDRESS_LIST=</td>
</tr>
<tr>
<td></td>
<td>(ADDRESS=</td>
</tr>
<tr>
<td></td>
<td>(PROTOCOL=IPC)</td>
</tr>
<tr>
<td></td>
<td>(KEY=SMF)</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td></td>
<td>(ADDRESS=</td>
</tr>
<tr>
<td></td>
<td>(PROTOCOL=tcp)</td>
</tr>
<tr>
<td></td>
<td>(HOST=marco)</td>
</tr>
<tr>
<td></td>
<td>(PORT=1521)</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>The ORACLE_SID for the SMF database is SMF. The listener can be made aware of this by adding an ADDRESS entry to the ADDRESS_LIST. The listener should also be instructed to listen for requests on a designated TCP port on the USMS machine.</td>
</tr>
<tr>
<td><strong>5</strong></td>
<td>Add an entry to the SID_LIST</td>
</tr>
<tr>
<td>Where:</td>
<td>• SID-NAME=SMF</td>
</tr>
<tr>
<td></td>
<td>• ORACLE_HOME=&lt;oracle version&gt;</td>
</tr>
<tr>
<td></td>
<td>• GLOBAL_DBNAME=SMF.&lt;Host name&gt;</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>The example below illustrates the configuration for an USMS machine called &quot;marco&quot;:</td>
</tr>
<tr>
<td></td>
<td>SID_LIST_LISTENER=(SID_LIST=</td>
</tr>
<tr>
<td></td>
<td>(SID_DESC=</td>
</tr>
<tr>
<td></td>
<td>(SID_NAME=SMF)</td>
</tr>
<tr>
<td></td>
<td>(ORACLE_HOME=/opt/oracle/product/7.3.3)</td>
</tr>
<tr>
<td></td>
<td>(GLOBAL_DBNAME=SMF.MARCO)</td>
</tr>
<tr>
<td></td>
<td>)</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>The listener also needs to know where it can find the information for any particular ORACLE_SID. This is accomplished via the SID_LIST. The listener needs to know the name of the SID, the Oracle home directory and the global database name.</td>
</tr>
<tr>
<td><strong>6</strong></td>
<td>Ensure the following settings are not modified:</td>
</tr>
<tr>
<td></td>
<td>STARTUP_WAIT_TIME_LISTENER = 0</td>
</tr>
<tr>
<td></td>
<td>CONNECT_TIMEOUT_LISTENER = 10</td>
</tr>
<tr>
<td><strong>7</strong></td>
<td>Delete the following entries:</td>
</tr>
<tr>
<td></td>
<td>USE_PLUG_AND_PLAY_LISTENER = TRUE</td>
</tr>
<tr>
<td></td>
<td>USE_CKPFILE_LISTENER = TRUE</td>
</tr>
<tr>
<td><strong>8</strong></td>
<td>Save and close the file.</td>
</tr>
</tbody>
</table>

*Continued on next page*
## Configuring the Oracle Listener, Continued

### Procedure (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>9</strong></td>
<td>Type <code>lsnrctl stop</code></td>
</tr>
<tr>
<td><strong>Result:</strong></td>
<td>This will stop the listener. At this stage, if everything worked correctly, you should no longer need to be the Oracle user.</td>
</tr>
</tbody>
</table>

| **10** | Ensure that you are the root user. |

| **11** | Edit the file `/IN/html/sms.html` |
| **Example command:** | `vi /IN/html/sms.html` |

| **12** | Edit the following parameters. |
| **Where:** | - HOST=HOST_IP_ADDR (address of the USMS machine)  
- DATABASEID=LPORT:DB_SID (listener port and its SID) |
| **Example:** | HOST=marco  
DATABASEID = 1521:SMF |
| **Result:** | This step changes the parameters in this file to reflect those of the listener. The installation process will have tried to automatically configure this file for you, but it may not be completely accurate. |
Configuring the SNMP Agent

Introduction

SNMP trap relaying is not automatically enabled. If you require SNMP trap relaying then you must perform the steps described in this topic.

The SNMP Agent supports the following functionality:

- forwarding of alarms as SNMP traps, using the Alarm Relay mechanism (see SMS User's Guide)
- resynchronisation of traps, enabling an SNMP manager to request resend of traps.

Traps may be forwarded to multiple SNMP managers.

Note: This is subject to the following restrictions:

- all managers must use the same port to receive SNMP traps;
- all managers must be configured to use the same Community string; and
- any triggering of the resynchronisation mechanism will result in duplicate traps being forwarded to all managers.

The SNMP agent is configured via the Alarm Notification screen and the snmp config file as described in this section. The config file is:

```
/IN/service_packages/SMS/etc/snmp.cfg
```

The name of the network management station is defined by the destination field in the rule, used to match alarms. This allows alarms to be sent to multiple machines and also to determine which alarms should be sent to which machines. The SNMP specific parameters are:

- TARGET = "SNMP"
- DESTINATION = <manager hostname>

The other parameters are the same for all destinations and are determined from this configuration file, read at the start up of the smsAlarmRelay program.

In understanding these parameters, you must be familiar with the Simple Network Management Protocol (SNMP).

We currently support SNMP v3 (IETF STD0062). SNMP v1 (IETF RFC1157) traps are supported for backward compatibility purposes only.

To support integration with as broad a range of SNMP managers as possible, two forms of SNMP trap are supported:

- Opaque traps include all of the fault data in a single structured data type; and
- Multiple variable traps, wherein each fault datum is represented by a distinct trap variable.

A single trap type must be chosen for each installation. See the "opaque" and "specific" configuration parameter descriptions below for details.

Continued on next page
Follow these steps to turn on SNMP relaying of alarms.

**Note:** Like any command line switches, the –p can appear at any point in the command line. –p is a parameter without any options, and is used to enable SNMP relaying of alarms. SNMP relaying of alarms is off by default.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Open the smsAlarmRelayStartup.sh script with a text editor.  
      | Example command: vi /IN/service_packages/SMS/etc/snmp.cfg |
| 2    | Add –p to the command line. |
| 3    | Save and close the file. |

This text shows the content of an example snmp.cfg file.

```
use-SNMPv3: 1
listenPort: 1161
userName: smf_oper
community: public
my-addr: addr
trap: 6
specific: 1
opaque: 1
port: 162
```

The parameters available in this file are described below. The only parameter that you are required to modify is “my-addr”; the rest are given for reference only.

**Note:** Separate the parameter from the value using the colon ‘:’.

```
use-SNMPv3
```

**Syntax:** use-SNMPv3: <0|1>

**Description:** The version of the SNMP implementation.

**Type:** Integer

**Optionality:** Mandatory

**Allowed:** 0 SNMPv1 is enabled  
               1 SNMPv3 is enabled

**Default:** 1

**Example:** use-SNMPv3: 1
Configuring the SNMP Agent, Continued

snmp.cfg file parameters

listenPort

Syntax: listenPort: <port>
Description: The UDP port number from which smsAlarmRelay listens for get- and set-variable requests.
Type: Integer
Optionality: Mandatory
Allowed: 1 - 65535
Notes: If use-SNMPv3 (on page 70) is set to 0, the listenPort parameter has no effect.
Example: listenPort: 1161

userName

Syntax: userName: <name>
Description: Used by smsAlarmRelay when it listens on a standard SNMP port that has already been opened.
Type: String
Optionality: Mandatory
Default: smf_oper
Notes: In order to open the standard SNMP port, smsAlarmRelay needs root privileges. Once the port is open, smsAlarmRelay's privileges are restricted to those assigned to <name>.

community

Syntax: community: <type>
Description: The community to which smsAlarmRelay belongs.
Type: String
Optionality: Mandatory
Default: public

Continued on next page
Configuring the SNMP Agent, Continued

snmp.cfg file parameters (continued)

my-addr

Syntax: my-addr: <addr>
Description: The Internet protocol address of the computer on which smsAlarmRelay is installed.
Type: String
Optionality: Mandatory
Allowed: May be either a symbolic host name or an Internet protocol number expressed in dotted-decimal format.
Notes: In SNMP terminology, <addr> is called agent-addr. Most hosts have at least two addresses, the second one being the loop-back address: 127.0.0.1.
Example: A symbolic host name might be SMS_main_1.
my-addr: SMS_main_1
An Internet protocol number could be 192.168.7.140.
my-addr: 192.168.7.140

trap

Syntax: trap: <int>
Description: The value of the generic trap.
Type: Integer
Optionality: Mandatory
Allowed: 6
Default: 6

specific

Syntax: specific: <int>
Description: An SNMP-specific trap parameter.
Type: Integer
Optionality: Optional
Allowed: 1 A single opaque binding.
2 Multiple variable bindings per trap, for each parameter.
3 A single opaque binding in x733 format.
4 Multiple x733 variable bindings per trap.
Notes: If you use the specific parameter, you must also set the opaque (on page 73) parameter.
Example: specific: 1

Continued on next page
opaque
Syntax: opaque: <flag>
Description: Defines encoding for SNMP specific traps.
Type: Integer
Optionality: Mandatory if the specific (on page 72) parameter is used.
Allowed: 0 Use if specific is set to 2 or 4.
1 Use if specific is set to 1 or 3.
Notes: The value for <flag> depends on the value assigned to the specific parameter.
Example: opaque: 1

port
Syntax: port: <port>
Description: The Internet Protocol port number of the remote SNMP manager computer.
Type: Integer
Optionality: Mandatory
Allowed: 1 - 65535
Default: 162
Notes: 162 is the SNMP trap port.

param-oid
Syntax: param-oid: <id>
Description: The alarm parameter argument assigned to Oracle.
Type: String
Optionality: Optional but deprecated.
Allowed: 1.2.36.52947743
Default: 1.2.36.52947743
Notes: The <id> is constructed from the values of the sub-parameters listed below.

<table>
<thead>
<tr>
<th>iso</th>
<th>country</th>
<th>australia</th>
<th>Oracle</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>36</td>
<td>52947743</td>
</tr>
</tbody>
</table>

my-oid
Syntax: my-oid: <id>
Description: The alarm parameter argument assigned to Oracle.
Type: String
Optionality: Optional but deprecated.
Allowed: 1.2.36.52947743
Default: 1.2.36.52947743

Continued on next page
Configuring the SNMP Agent, Continued

snmp.cfg file parameters (continued)

notification-oid

Syntax: notification-oid: <str>

Description: A variable that can be queried or changed remotely.

Type: String

Optionality: Optional but deprecated.

Allowed: Constructed from the value of the param-oid (on page 73) parameter to which is appended two additional digits. The value of each digit is determined by the format of alarms.

- 1.2.36.52947743.1.1: Opaque encoding of Oracle fields.
- 1.2.36.52947743.1.2: id, .3 = machine, .4 = time, .5 = cpu, etc.
- 1.2.36.52947743.2.1: Opaque encoding of X.733 fields.
- 1.2.36.52947743.2.2: Managed object instance, .3 event type, etc.

Default: 1.2.36.52947743.2.1

Notes: The notification-oid parameter requires that:
- The use-SNMPv3 (on page 70) parameter is set for SNMP version 3.
- The listenPort (on page 71) parameter is configured.

The format of SNMP messages is defined in IETF STD0062.

At the top level the “Message” element has the “version” field set in accordance with the SNMP version set by the “use-SNMPv3” configuration parameter. The rest of the formatting differs according to the SNMP version that is being used.

SNMP v1

The SNMP v1 message is built up from each line of this table.

<table>
<thead>
<tr>
<th>Part</th>
<th>Set from</th>
</tr>
</thead>
<tbody>
<tr>
<td>version</td>
<td>Set by the use-SNMPv3 configuration parameter.</td>
</tr>
<tr>
<td>community</td>
<td>Set via the community configuration parameter.</td>
</tr>
<tr>
<td>enterprise</td>
<td>Set using the my-oid configuration parameter.</td>
</tr>
<tr>
<td>agent-addr</td>
<td>The IP address of the USMS set using my-addr parameter.</td>
</tr>
<tr>
<td>generic-trap</td>
<td>Set using the trap configuration parameter.</td>
</tr>
<tr>
<td>specific-trap</td>
<td>Set using the the specific parameter.</td>
</tr>
</tbody>
</table>

Continued on next page
Configuring the SNMP Agent, Continued

Formatting as an SNMP trap message (continued)

SNMP v3
The SNMP v3 message is built up as follows.

- version - set by the use-SNMPv3 configuration parameter
- Global Header - including a usm security model
- security parameters
  - authoritative Engine ID - security ID
  - engine boots - record of the number of boots of the alarmRelay
  - engine time - record of the up of the alarmRelay
- context engine ID - PID of smsAlarmRelay
- context name - "smsAlarmRelay"
- v2 trap PDU
  - error status
  - error index

variable bindings
The "variable-bindings" take one of two forms, in accordance with the settings of the "opaque" and "specific" configuration parameters.

The opaque form is composed of a sequence containing a single item. That single item is itself a sequence comprising of a pair. The pair is the object id of the alarm (obtained from the configuration file) and the alarm data itself encased as an "Opaque" data item.

The multiple variable form is composed of a sequence of pairs, each pair being an object id identifying the variable and the variable values. The object ids and variable datatypes are specified in the MIB.

See The SMF AlarmMessage Format for the ASN.1 format of the alarm data.

Transmission of the SNMP trap message
Given the trap message that has been previously formatted we can now send it to the network management station. As defined in RFC 1157, the message is sent over the User Datagram Protocol (UDP). The destination IP address and the port are specified in the configuration file.

Failure to send the trap does not raise an alarm as this would lead to an infinite loop of alarm messages.

Starting and stopping
The SNMP additions to the smsAlarmRelay will send an "start" trap to all configured destinations when it starts up. Similarly, it will send a "stopped" trap and process shutdown.

Continued on next page
Configuring the SNMP Agent, Continued

By default, SNMP trap relaying is not performed. Therefore the smsAlarmRelayStartup.sh script must be edited and the `smsAlarmRelay` (on page 99) process restarted using the steps below.

Follow these steps to restart the smsAlarmRelay daemon.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Type following command to find the process ID: `ps -ef</td>
</tr>
<tr>
<td>2</td>
<td>Type <code>kill -TERM &lt;pid&gt;</code>&lt;br&gt;<strong>Result:</strong> The process is terminated and will be restarted by the init tab process.</td>
</tr>
</tbody>
</table>
SMF AlarmMessage Format

Introduction

This topic provides the format of the SMFAlarmMessage including the MIB definitions.

Alarm Table fields

This table defines the layout of the SMF_ALARM_MESSAGE and SMF_ALARM_DEFN tables in the SMF from which the alarms are derived.

<table>
<thead>
<tr>
<th>Name</th>
<th>Field size</th>
<th>Field type</th>
<th>Null value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>38</td>
<td>NUMBER</td>
<td>not null</td>
</tr>
<tr>
<td>machine</td>
<td>16</td>
<td>VARCHAR2</td>
<td>not null</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(15 characters for hostname; 1 terminating character)</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>DATE</td>
<td>not null</td>
<td></td>
</tr>
<tr>
<td>cpu</td>
<td>3</td>
<td>NUMBER</td>
<td>not null</td>
</tr>
<tr>
<td>name</td>
<td>6</td>
<td>NUMBER</td>
<td>not null</td>
</tr>
<tr>
<td>subsystem</td>
<td>24</td>
<td>VARCHAR2</td>
<td>not null</td>
</tr>
<tr>
<td>severity</td>
<td>1</td>
<td>NUMBER</td>
<td>not null</td>
</tr>
<tr>
<td>description</td>
<td>256</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>opcomment</td>
<td>256</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>count</td>
<td>4</td>
<td>NUMBER</td>
<td>not null</td>
</tr>
<tr>
<td>close_time</td>
<td>DATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>status</td>
<td>7</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>change_sequence</td>
<td>38</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>managed_object_instance</td>
<td>2000</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>event_type</td>
<td>2</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>probable_cause</td>
<td>4</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>specific_problem</td>
<td>256</td>
<td>VARCHAR2</td>
<td></td>
</tr>
<tr>
<td>perceived_severity</td>
<td>1</td>
<td>NUMBER</td>
<td></td>
</tr>
<tr>
<td>additional_text</td>
<td>1000</td>
<td>VARCHAR2</td>
<td></td>
</tr>
</tbody>
</table>

MIB field mappings - SMF_ALARM_MESSAGE

This table provides the SMF_ALARM_MESSAGE to MIB field mappings.

<table>
<thead>
<tr>
<th>DB Alarm</th>
<th>MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>id Mapped directly (unique ID)</td>
</tr>
<tr>
<td>1</td>
<td>machine Mapped directly (hostname)</td>
</tr>
<tr>
<td>2</td>
<td>time Mapped directly (&quot;YYYYMMDDHHMMSS&quot;)</td>
</tr>
<tr>
<td>3</td>
<td>cpu Mapped directly (CPU number)</td>
</tr>
<tr>
<td>4</td>
<td>name = 0 for Solaris &amp; HPUX</td>
</tr>
<tr>
<td>6</td>
<td>subsystem Mapped directly (process identifier)</td>
</tr>
</tbody>
</table>
## SMF AlarmMessage Format, Continued

### MIB field mappings - SMF_ALARM_MESSAGE (continued)

<table>
<thead>
<tr>
<th>DB Alarm</th>
<th>MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 severity</td>
<td>Mapped directly (0=NOTICE, 2=WARNING, 4=ERROR, 6=CRITICAL, 8=CLEARANCE)</td>
</tr>
<tr>
<td>8 description</td>
<td>Mapped directly (free text)</td>
</tr>
<tr>
<td>9 opcomment</td>
<td>Mapped directly (free text)</td>
</tr>
<tr>
<td>10 count</td>
<td>Mapped directly (number of duplicates)</td>
</tr>
<tr>
<td>close_time</td>
<td>Not sent</td>
</tr>
<tr>
<td>5 status</td>
<td>Mapped directly (&quot;OPEN&quot;, &quot;PENDING&quot;, &quot;CLOSED&quot;)</td>
</tr>
<tr>
<td>change_sequence</td>
<td>Not sent</td>
</tr>
</tbody>
</table>

This table provides the SMF_ALARM_DEFN to MIB field mappings.

<table>
<thead>
<tr>
<th>DB Alarm</th>
<th>MIB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alarm_type_id</td>
<td>not sent</td>
</tr>
<tr>
<td>event_type</td>
<td>Mapped directly (event_type)</td>
</tr>
<tr>
<td>probable_cause</td>
<td>Mapped directly (probable_cause)</td>
</tr>
<tr>
<td>severity</td>
<td>Mapped directly (severity)</td>
</tr>
<tr>
<td>specific_problem</td>
<td>Mapped directly (specific_problem)</td>
</tr>
<tr>
<td>recommended_action</td>
<td>not sent</td>
</tr>
<tr>
<td>additional_text</td>
<td>Prefixed with description and mapped to additional_text</td>
</tr>
<tr>
<td>present_to_am</td>
<td>not sent</td>
</tr>
<tr>
<td>present_to_ar</td>
<td>not sent</td>
</tr>
<tr>
<td>autoclear_period</td>
<td>not sent</td>
</tr>
<tr>
<td>regular_expression</td>
<td>not sent</td>
</tr>
<tr>
<td>notes</td>
<td>not sent</td>
</tr>
</tbody>
</table>

### SMF Listen Messages

An SNMP manager may trigger the resend of traps by setting eServDataLastChgSeq to the value of the identifier (id or eServId) of the last successfully received trap.

**Note:** Use of this mechanism will cause traps to be sent to all active SNMP managers.
Defining the Screen Language

Introduction
The Default language file sets the language which the Java Administration screens will start in. The user can change to another language once they have logged in.

The default language can be changed by the System Administrator.

By default, the language is set to English. If English is your preferred language, you can skip this step and proceed to the next configuration task: Defining the Help Screen Language.

Default.lang
When SMS is installed, a file called Default.lang is created in the application's language directory in the screens module. This contains a soft-link to the language file which defines the language which will be used by the screens.

If a Default.lang file is not present, the English.lang file will be used.

The SMS Default.lang file is:
/IN/html/SMS/language/Default.lang

Example screen language
If Dutch is the language you want to set as the default, create a soft-link from the Default.lang file to the Dutch.lang file.

User-specific language settings
All screens in the SMS are able to support selected languages. On login, the screens are displayed in the default language. Languages may then be selected for a user in the Configuration field of the User Management screen. Once a language is selected for a user, it is stored in their profile.

For more information about setting the Configuration field, see SMS User's Guide.

Procedure
Follow these steps to set the default language for your SMS Java Administration screens.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the following directory: 
      | /IN/html/SMS/language |
|      | Example command: cd /IN/html/SMS/language/ |
| 2    | Ensure the Default.lang file exists in this directory. |
| 3    | If the required file does not exist, create an empty file called Default.lang. |
| 4    | Ensure that the language file for your language exists in this directory. 
The file should be in the format: 
<language>.lang 
Where: 
<language> = your language. 
Example:  
Spanish.lang |

Continued on next page
Defining the Screen Language, Continued

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 5    | If the required language file does not exist, either:  
      • create a new one with your language preferences, or  
      • contact Oracle support.  
      To create a language file, you will need a list of the phrases and words used in the screens. These should appear in a list with the translated phrase in the following format:  
      original phrase=translated phrase  
      Any existing language file should have the full set of phrases. If you do not have an existing file to work from, contact Oracle support with details. |
| 6    | Create a soft link between the Default.lang file, and the language file you want to use as the default language for the SMS Java Administration screens.  
      **Example command:** `ln -s Dutch.lang Default.lang` |
Defining the Help Screen Language

Introduction
The Default Helpset file sets the language which the help system for the Java Administration screens will start in. The user can change to another language once they have logged in.

The default language can be changed by the System Administrator. By default, the language is set to English.

Default.SMS.hs
When SMS is installed, a file called Default.SMS.hs is created in the application's language directory in the screens module. This contains a soft-link to the language file which defines the language which will be used by the screens.

If a Default.SMS.hs file is not present, the English.SMS.hs file will be used.

If a Default.SMS.hs file is present, a user must explicitly set their language to their required language in the Tools screen or the default language will be used.

The Default.SMS.hs file is:
/IN/html/SMS/helptext/Default.SMS.hs

Example helpset language
If Dutch is the language you want to set as the default, create a soft-link from the Default.SMS.hs file to the Dutch.SMS.hs file.

Procedure
Follow these steps to set the default language for your SMS Java Administration screens.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the following directory:  
      | /IN/html/SMS/helptext  
      | **Example command:** cd /IN/html/SMS/helptext |
| 2    | Ensure the Default.SMS.hs file exists in this directory. |
| 3    | If the required file does not exist, create an empty file called Default.SMS.hs. |
| 4    | Ensure that the language file for your language exists in this directory.  
      | The file should be in the format:  
      | `<language>.SMS.hs`  
      | **Where:**  
      | `<language>` = your language.  
      | **Example:**  
      | Dutch.SMS.hs |
| 5    | If the required language file does not exist, either:  
      | • create a new one with your language preferences, or  
      | • contact Oracle support.  
      | To create a language file, you will need a list of the phrases and words used in the screens. These should appear in a list with the translated phrase in the following format:  
      | `original phrase=translated phrase`  
      | Any existing language file should have the full set of phrases. If you do not have an existing file to work from, contact Oracle support with details. |

Continued on next page
## Defining the Help Screen Language, Continued

**Procedure** (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 6    | Create a soft link between the Default.SMS.hs file, and the language file you want to use as the default language for the SMS Java Administration screens.  
  **Example command:** `ln -s Dutch.Acs_Service.hs Default.Acs_Service.hs` |
Setting up the screens

Introduction

On first use of the application, the screens need to be set up on your local machine. Use an internet browser to open the relevant webpage:

- To run the screens through the SMS application.
  
  http://<smshostname>/sms.html

Where:

<smshostname> is the hostname of an USMS in the IN.

Result: The SMS Java Administration screens will start.
Customising the screens

The following parameters are available to customise the screen display. Parameters are added to the acs.html or the sms.html in the following format:

```html
<param name=<parameter> value="<value>">
```

Where:

- `<parameter>` is the parameter from the list below.
- `<value>` is the value that the parameter is to be set to.

**databaseID**

- **Syntax:** `databaseID value = "<port>:<sid>"
- **Description:** The SQLNET port to use to connect to the database, and the database SID.
- **Type:** String
- **Optionality:** Mandatory
- **Example:** `databaseID value = "1234:SMF"

**trace**

- **Syntax:** `trace value = <bool>`
- **Description:** Whether tracing should be enabled for the xerlin-based Configuration Editor. The output is displayed in the Java console window, if enabled.
- **Type:** Boolean
- **Optionality:** Optional (default used if not set).
- **Allowed:** `<on | off>, <true | false>, <yes | no>, <1 | 0>, <enabled | disabled>`
- **Default:** off
- **Example:** `trace value = "ON"

**databaseHost**

- **Syntax:** `databaseHost value = "<ip>:<port>:<sid>"
- **Description:** The IP address and port to use to connect to the SMF database, and the database SID.
- **Type:** String
- **Optionality:** Optional (if not set, databaseID parameter is used to connect)
- **Example:** `databaseHost value = "192.0.2.0:1234:SMF"

Continued on next page
Customising the screens, Continued

Introduction (continued)

clusterDatabaseHost

Syntax: See Example html applet (on page 87).
Description: This specifies a connection string including a host and an alternative host address, in case the first IP address is unavailable.
Type: String
Optionality: Optional
Example: clusterDatabaseHost value = "(DESCRIPTION=(LOAD_BALANCE=YES)(FAILOVER=ON)(ENABLE=BLACK)
(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=10.10.100.10)(PORT=1234))
(ADDRESS=(PROTOCOL=TCP)(HOST=10.11.111.00)(PORT=5678)))
(CONNECT_DATA=(SERVICE_NAME=SMF)(FAILOVER_MODE=(TYPE=SESSION)(METHOD=BASIC)(RETRIES=5)(DELAY=3)))"

namingServerPort

Syntax: namingServerPort = \<port>\nDescription: The port to talk to the naming server on.
Type: Integer
Optionality: Optional
Default: 5556
Example: namingServerPort = "5556"

logo

Syntax: See Example html applet (on page 87).
Description: Use to display different system graphics than the system is installed with.
Type: String
Optionality: Optional
Allowed: A valid network path.
Default: None
Notes: Use this parameter and set the value to the relative pathname to the substitute gif files.
Example: <param name=logo value="/IN/html/Acs_Service/images">

Continued on next page
Customising the screens, Continued

TZ
Syntax: <param name=TZ value="<TZ>">
Description: The screens will display all time and date values in the Timezone set by this parameter.
Type: String
Optionality: Optional (default used if not set)
Allowed: Any Java Supported timezone.
Default: GMT
Notes: For a full list a Java Supported timezones see the ACS Technical Guide - Appendix TimeZones.
Example: See Example html applet (on page 87).

webdavPort
Syntax: webdavPort= <port>
Description: Used by the Xerlin-based Configuration Editor for performing remote file operations.
Type: Integer
Optionality: Optional (default used if not set).
Default: 80
Example: webdavPort = 80

webdavDebug
Syntax: webdavDebug = <true|false>
Description: Whether debug should be enabled for the xerlin-based Configuration Editor.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true enabled (The output is displayed in the Java console window.)
false disabled
Default: true
Example: webdavDebug = true

configPlugin
Syntax: configPlugin = "~<path>/<plugin>"
Description: Defines information used during startup.
Type: String
Optionality: Optional (default used if not set).
Default: /Configuration/plugins/exampleConfig/plugin.xml
Example: configPlugin = "/Configuration/plugins/exampleConfig/plugin.xml"
Customising the screens, Continued

**Introduction (continued)**

```
workingConfigFile
Syntax:  workingConfigFile = "~<path>/<plugin>"
Description: This is the location of the working XML configuration file.
Type: String
Optionality: Optional (default used if not set).
Default: /Configuration/xml/exampleConfigWorking.xml
Example: workingConfigFile = "/Configuration/xml/exampleConfigWorking.xml"
```

```
masterConfigFile
Syntax:  masterConfigFile = "~<path>/<plugin>"
Description: This is the location of the master XML configuration file.
Type: String
Optionality: Optional (default used if not set).
Default: /Configuration/xml/exampleConfigMaster.xml
Notes: This filename cannot be accessed using the Xerlin editor menu options, "Save As" or "Webdav Save As". Xerlin access to this file is prohibited.
Example: masterConfigFile = "/Configuration/xml/exampleConfigMaster.xml"
```

### Example html applet

The following is an example of how to configure the sms.html files applet parameters.

```html
<applet
  codebase = "."
  archive  = 
    "sms.jar.sig,common.jar.sig,ojdbc14.jar.sig,Xerlin-
     1.3.jar.sig,xerlin_libs.jar.sig,jhbasics.jar.sig,acs.jar.sig,jch
     art.jar.sig,ccs.jar.sig"
  code     = "UserScreens.Applet1.class"
  name     = "SMSApplet"
  alt      = "Please install/enable Java Runtime Plugin to run
     SMS"
  width    = 0
  height   = 0
  hspace   = 0
  vspace   = 0
  align    = Middle>
  <param name=TZ value="GMT">
  <param name=host value="123.123.12.12">  
  <param name=logo value="SMS/images/Oracle.gif">
  <param name=databaseID value="1521:SMF">
  <param name=INProtocol value="IN_PROTOCOL">
  <param name=UseAnnouncements value="YES">
  <param name=masterConfigFile
     value="/Configuration/plugins/exampleConfig/plugin.xml">
  <param name=workingConfigFile
     value="/Configuration/xml/exampleConfigWorking.xml">
  <param name=masterConfigFile
     value="/Configuration/xml/exampleConfigMaster.xml">

Continued on next page
Customising the screens, Continued

Example html applet (continued)

```html
<param name=showEFM value=1>
<PARAM NAME=SuppressTagID VALUE="TRUE">
<PARAM NAME=Profile1 VALUE="-">
<PARAM NAME=Profile2 VALUE="-">
<PARAM NAME=ORB_HOST VALUE="prod-smp01">
<PARAM NAME=maximiseAcsScreens VALUE="false">
<PARAM NAME=BeORBTimeoutms VALUE="5000">

</APPLET>
```
Configuring Nodes

**USMS Nodes**
During installation of the SMS software, each USMS will be set-up so that it is a valid replication node. You should check that each node has at least the following configuration details:

- Valid primary address (or hostname)
- Node number of 1-16 (starting at 1), and
- Validator check box checked.

You can check the set-up via the Node Management screen in the SMS Administration screens. For more information on node configuration and set up see the *SMS User's Guide*.

---

**UAS Nodes**
In a clustered installation, each UAS has one Node Number associated with it:

- One in the range 256 to 511 for the Update Loader

These Node Numbers can be assigned using the *Node Management* screen in the SMS Java screens.

Each Update Loader should at least have:

- Valid primary address (or hostname)
- Node number in the range 256 to 511 (the Node Numbers of the Update Loader should start at 301).
- Empty validator check box.

For more information on node configuration and set up see the *SMS User's Guide*.

---

**Statistics nodes**
You must complete the process by configuring Statistics within the SMS, see *SMS User's Guide*.
# Installing Additional Applications

Follow these steps to install the applications.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Install each application to create a set of replication groups.</td>
</tr>
<tr>
<td>2</td>
<td>Decide which UASs will run this application.</td>
</tr>
<tr>
<td>3</td>
<td>Target all of the new groups (or such different set as is advised in the instructions for the application) onto each of these UASs (the i+256 node).</td>
</tr>
</tbody>
</table>

Please note that the order in which replication tables are added is important.
Chapter 6

Background Processes on the USMS

Overview

Introduction

This chapter provides a description of the programs or executables used by the System as background processes on an USMS.

Executable are located in the /IN/service_packages/SMS/bin directory.

Some executables have accompanying scripts that run the executables after performing certain cleanup functions. All scripts should be located in the same directory as the executable.

For more information about the processes and systems that use these programs and executables, see System Overview (on page 1).

Important: It is a pre-requisite for managing these core service functions that the operator is familiar with the basics of Unix process scheduling and management. Specifically, the following Unix commands:

- init (and inittab)
- cron (and crontab)
- ps
- kill

This chapter contains the following topics:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmnConfigRead</td>
<td>92</td>
</tr>
<tr>
<td>cmnReceiveFiles</td>
<td>93</td>
</tr>
<tr>
<td>smsAlarmDaemon</td>
<td>94</td>
</tr>
<tr>
<td>smsAlarmManager</td>
<td>97</td>
</tr>
<tr>
<td>smsAlarmRelay</td>
<td>99</td>
</tr>
<tr>
<td>smsConfigDaemon</td>
<td>102</td>
</tr>
<tr>
<td>smsConfigDaemonScript</td>
<td>103</td>
</tr>
<tr>
<td>smsCdrProcess.sh</td>
<td>105</td>
</tr>
<tr>
<td>smsDbCleanup.sh</td>
<td>106</td>
</tr>
<tr>
<td>smsLogCleaner</td>
<td>107</td>
</tr>
<tr>
<td>smsMergeDaemon</td>
<td>109</td>
</tr>
<tr>
<td>smsMaster</td>
<td>110</td>
</tr>
<tr>
<td>smsNamingServer</td>
<td>111</td>
</tr>
<tr>
<td>smsReportsDaemon</td>
<td>112</td>
</tr>
<tr>
<td>smsReportScheduler</td>
<td>114</td>
</tr>
<tr>
<td>smsReportCleanupStartup.sh</td>
<td>116</td>
</tr>
<tr>
<td>smsStatsDaemon</td>
<td>117</td>
</tr>
<tr>
<td>smsStatsThreshold</td>
<td>118</td>
</tr>
<tr>
<td>smsSendConfig.sh</td>
<td>119</td>
</tr>
<tr>
<td>smsTaskAgent</td>
<td>120</td>
</tr>
<tr>
<td>smsTrigDaemon</td>
<td>123</td>
</tr>
</tbody>
</table>
## cmnConfigRead

### Purpose

cmnConfigRead is used by the installation process to read the configuration files. cmnConfigRead reads the Oracle configuration file (eserv.config), specified by the Oracle_CONFIG_FILE environment variable and returns the value of `path`. This can be used in commands to return the eserv.config specified path value.

**Example:**

```bash
FILENAME=`cmnConfigRead CCS.MyReport.filename /IN/service_packages/CCS/tmp/MyReport.log`
```

sets `FILENAME` to the value of `CCS.MyReport.filename`. If `CCS.MyReport.filename` is not present or there is an error, `FILENAME` will default to `/IN/service_packages/CCS/tmp/MyReport.log`.

### Startup

cmnConfigRead is started by the system and is not intended to be changed by the user.
<table>
<thead>
<tr>
<th>Purpose</th>
<th>cmnReceiveFiles collects EDRs from cmnPushFiles and writes them to the specified directory on the USMS.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>cmnReceiveFiles is started by the following entry in /etc/inetd.conf:</td>
</tr>
<tr>
<td></td>
<td>smsoperFile stream tcp nowait smf_oper /IN/service_packages/SMS/bin/cmnReceiveFilesStartup.sh cmnReceiveFilesStartup.sh</td>
</tr>
<tr>
<td>Parameters</td>
<td>cmnReceiveFiles does not have any direct parameters or configuration. Most details are provided by cmnPushFiles with the EDR.</td>
</tr>
<tr>
<td></td>
<td>The port cmnReceiveFiles listens on is set in /etc/services in the following line:</td>
</tr>
<tr>
<td></td>
<td>smsoperFile  2028/tcp # cmnAddInetServicesEntry</td>
</tr>
<tr>
<td>Important:</td>
<td>The port number must match the port specified by cmnPushFiles.</td>
</tr>
<tr>
<td>Failure</td>
<td>If cmnReceiveFiles fails, the EDRs will stay on the UAS and be moved to the retry directory. For more information about this process, see cmnPushFiles (on page 131).</td>
</tr>
<tr>
<td>Output</td>
<td>cmnReceiveFiles writes the EDRs to the directory specified by cmnPushFiles.</td>
</tr>
</tbody>
</table>
smsAlarmDaemon

**Purpose**

The smsAlarmDaemon runs on all alarm-managed nodes in the SMS system, including the USMS node itself. The role of the smsAlarmDaemon is to gather alarms from the following sources:

- System error log (/var/adm/syslog.log or /var/log/syslog)
- Oracle error log ($ORACLE_BASE/admin/<SID>/bdump/alert_<SID>.log)
- Sigtran stack logs (/IN/service_packages/SLEE/stats) [If installed]

On the USMS machine itself, the resultant error messages are written directly into the SMF_ALARM_MESSAGE table in the database. When run on other nodes, replication is used to update the SMF_ALARM_MESSAGE table.

**Alarm replication and buffering**

The smsAlarmDaemon allows only a limited number of alarms to be sent within a configured time period. Both the number of messages that can be sent within a time period and the length of each period can be configured from the command line.

If more messages arrive then are allowed through the filter, the remaining messages will be buffered and sent at a later time. The buffer size is limited but can hold a large number of messages. If it needs to make more space, it will discard messages of the lowest severity (informational). The buffer also has an upper limit, ensuring that the daemons will not grow unchecked. This upper limit will default to a maximum of 1000 messages and can be configured.

If more than one of the same alarm appears within the configured time, only one update request is sent.

**Startup**

In an unclustered install, this task is started by entry sms5 in the inittab, via the shell script:

```
/IN/service_packages/SMS/bin/smsAlarmDaemonSmsStartup.sh
```

In a clustered install, this task is started by the clustering software, via the shell script:

```
/IN/service_packages/SMS/bin/smsAlarmDaemonCluster.sh
```

**Configuration**

The smsAlarmDaemon accepts the following command line arguments.

**Usage:**

```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a &lt;path&gt;</td>
<td>Null</td>
<td>Location of the Oracle Alert Log. By default no oracle alert log will be generated.</td>
</tr>
<tr>
<td>-c</td>
<td>1</td>
<td>Commit Rate - number of inserts before committing to the database.</td>
</tr>
<tr>
<td>-d</td>
<td>Sort messages</td>
<td>Disable sorting of messages by severity in buffer. Specifically, messages are kept in the buffer and subsequently written into the SMF database, in the same sequence in which they are received.</td>
</tr>
</tbody>
</table>

Continued on next page
### Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-f</td>
<td>No filtering</td>
<td>Filtering - delete duplicate alarms and increase the alarm count.</td>
</tr>
<tr>
<td>-g</td>
<td>Uses local time</td>
<td>GMT timezone - use GMT instead of local time.</td>
</tr>
<tr>
<td>-h &lt;secs&gt;</td>
<td>60</td>
<td>Heartbeat message. Will be forced to be greater or equal to time period (seconds).</td>
</tr>
<tr>
<td>-i</td>
<td>Use fuzzy matching</td>
<td>Filtering type - use exact matching (rather than fuzzy matching). Indicates that duplicate matches should be performed on text only (that is, excluding digits).</td>
</tr>
<tr>
<td>-l &lt;secs&gt;</td>
<td>2</td>
<td>Filter Period - duration between linked-list checks (seconds).</td>
</tr>
<tr>
<td>-n &lt;number&gt;</td>
<td>5</td>
<td>Filter Number - number of alarm messages allowed within the time period.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong> integers</td>
</tr>
<tr>
<td>-m &lt;max&gt;</td>
<td>1000</td>
<td>Maximum number of alarm messages to buffer.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong> integers 1-1000000</td>
</tr>
<tr>
<td>-p</td>
<td>Do not drop messages</td>
<td>Drop low priority messages when buffer is full. Specifically, when -m &lt;max&gt; messages have been received but it is not yet time to write the buffer contents to the &lt;box_SMF_ac&gt; database, then low priority messages in the buffer are dropped in favour of higher priority messages that may be received on its input stream.</td>
</tr>
<tr>
<td>-r &lt;node&gt;</td>
<td>Direct to the Oracle DB</td>
<td>Rep Node - specify replication requester node.</td>
</tr>
<tr>
<td>-t &lt;secs&gt;</td>
<td>1</td>
<td>Commit Interval - maximum interval between committing to the database (seconds).</td>
</tr>
<tr>
<td>-u &lt;user/pass&gt;</td>
<td>/</td>
<td>Use the supplied Oracle user/password pair.</td>
</tr>
</tbody>
</table>

### Usage examples

Here is an example of using smsAlarmDaemon:

---

Continued on next page
### Usage examples (continued)

```bash
smsAlarmDaemon -l 5 -h 30 -n 10 -m 2000 -p -d -a /volB/home/saich -r 750 -u
smf/smf -f -i -g -c 2 -t 2
```

- Filter Period (-I) = 5 seconds
- Heart beat (-h) = Yes every 30 seconds
- Filter Number (-n) = 10 each period
- Max number(-m) = 2000 records
- Drop low priority messages (-p) = true
- Sort messages by severity (-d) = false
- Oracle Alert Log location (-a) = /volB/home/saich
- Rep node (-r) = 750
- Oracle User (-u) = smf/smf
- Filtering (-f) = Multiple alarms combined
- Filtering type (-i) = Exact match
- GMT timezone (-g) = Yes
- Commit Rate (-c) = every 2 number of inserts
- Commit Interval (-t) = every 2 seconds if 2 records not reached

### Failure

The smsAlarmDaemon on each alarm-managed node in the installation will by default generate a health-check alarm once per minute. These health check alarms will be relayed in the same fashion as all other alarms.

If these health check alarms are not received at the target destination, then the smsAlarmDaemon may have failed, and should be investigated.

### Output

The smsAlarmDaemon writes error messages to the system messages file, and also writes additional output to:

```
/IN/service_packages/SMS/tmp/smsAlarmDaemonSms.log
```
smsAlarmManager

**Purpose**
The smsAlarmManager runs on the USMS. The role of the smsAlarmManager is to:
- Match alarm instances to the correct alarm types
- Automatically time out alarms that have not been cleared
- Record alarm instances that have no alarm type match

**Startup**
This task is started by entry efm1 in the inittab, via the shell script:

```
/IN/service_packages/EFM/bin/smsAlarmManagerStartup.sh
```

The inittab entry will be similar to that shown below:

```
efm1:34:respawn:su - smf_oper -c "exec 
/IN/service_packages/EFM/bin/smsAlarmManagerStartup.sh >> /IN 
/service_packages/EFM/tmp/smsAlarmManager.log 2>&1" > /dev/null
```

**Configuration**
The smsAlarmManager accepts the following command line arguments.

**Usage:**

```
smsAlarmManager -a <alarm_batch_size> -c <correlate_batch_size> -o 
<timeout_commit_rate> -p <pending_timeout_length> -r 
<reload_defn_interval> -s <number> -t <timeout_check_interval> -u 
<username / password>
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a &lt;alarm_batch_size&gt;</td>
<td>20</td>
<td>The number of alarms to attempt to find an ID for before carrying on to other tasks.</td>
</tr>
<tr>
<td>-c &lt;correlate_batch_size&gt;</td>
<td>20</td>
<td>The number of non-correlated CLEAR alarms to attempt to correlate against open alarms before carrying on to other tasks.</td>
</tr>
<tr>
<td>-o &lt;timeout_commit_rate&gt;</td>
<td>1000</td>
<td>The number of rows to update with automatic timeout before committing.</td>
</tr>
<tr>
<td>-p &lt;pending_timeout_length&gt;</td>
<td>280</td>
<td>The amount of time given to another node in the cluster before assuming that it has failed to generate an ALARM_TYPE_ID</td>
</tr>
<tr>
<td>-r &lt;reload_defn_interval&gt;</td>
<td>86400</td>
<td>interval (s) for reloading the alarm definitions for the DB.</td>
</tr>
<tr>
<td>-s</td>
<td>50000</td>
<td>interval (microseconds) to sleep for when no work to do</td>
</tr>
</tbody>
</table>

Continued on next page
smsAlarmManager, Continued

Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t</td>
<td>300</td>
<td>Interval between checks for alarms that need to be closed with a timeout.</td>
</tr>
<tr>
<td>&lt;timeout_check_interval&gt;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-u &lt;username / password&gt;</td>
<td>/</td>
<td>The username/password combination used to log into the DB. The default value is sufficient if smsAlarmManager is executed from the smf_oper user account</td>
</tr>
</tbody>
</table>

smsAlarmManager will respond to SIGHUP, to reread the regular expressions from the database.

Failure

The smsAlarmManager matches alarm instances with alarm types and updates the alarm instances with the extra information. Should any of the following occur the smsAlarmManager may have failed, and should be investigated.

- alarms missing expected information
- alarm clearances are not not being matched with the corresponding alarms
- alarms not being automatically timed out

If the smsAlarmManager cannot match an alarm instance with an alarm type, it will save the alarm text into the SMF_ALARM_UNKNOWN database table.

Output

On startup the smsAlarmManager logs the following information:

```
smsAlarmManager startup.
  Alarm Batch Size = 20
  Correlate Batch Size = 20
  Pending Timeout Length = 280
  Timeout Check Interval = 300
  Reload Defn Interval = 86400
  Timeout Commit Rate = 1000
  Sleep Time (microseconds) for no Work = 50000
  Username/Password = /
Aug 30 15:31:07 smsAlarmManager(18347) NOTICE: smsAlarmManager started.
Cache successfully reloaded
```
The smsAlarmRelay is responsible for implementing the **SNMP Agent** (on page 69). It runs continuously, polling the database to check for new entries written into the SMF_ALARM_MESSAGE table by the smsAlarmDaemon processes running on the various managed nodes which form the SMS-managed installation.

The information in the SMF_ALARM_MESSAGE is relayed to the destinations, as configured in the SMF_ALARM_HANDLER table using the Alarm Notification screens. For more information about how to configure alarm relay destinations, see the **SMS User’s Guide**.

You can configure smsAlarmRelay to do the following:

- send X.733 information with all forwarded alarms
- check for SNMP requests (to resend alarms)
- send version 3 (instead of version 1) SNMP traps

In an unclustered installation, this task is started by entry sms1 in the inittab, via the shell script:

/IN/service_packages/SMS/bin/smsAlarmRelayStartup.sh

In a clustered installation, this task is started by the cluster software, via the shell script:

/IN/service_packages/SMS/bin/smsAlarmRelayCluster.sh

The **smsAlarmRelay** accepts the following command line arguments.

**Usage:**

`smsAlarmRelay [-u <usr/pwd>] [-s <secs>] [-p] [-x] [-t]`

**Note:** SNMP processing is not currently enabled by default.

The available parameters are:

- **-u <usr/pwd>**
  
  **Syntax:**   
  
  `u <usr/pwd>`
  
  **Description:** The Oracle user and password pair.
  
  **Type:** String
  
  **Optionality:** Optional (default used if not set).
  
  **Default:** /

- **-s <secs>**
  
  **Syntax:**   
  
  `-s <secs>`
  
  **Description:** The number of seconds to sleep between database checks.
  
  **Type:** Integer
  
  **Optionality:** Optional (default used if not set).
  
  **Default:** 1
smsAlarmRelay, Continued

Parameters (continued)

-\(p\)

Syntax: \(-p\)
Description: Whether to do SNMP processing or not.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: set Use SNMP
not set Do not use SNMP
Default: not set

-\(x\)

Syntax: \(-x\)
Description: Whether to send SNMP traps in X.733 format or not.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: set Send in X.733 format.
not set Do not send in X.733 format.
Default: not set

-\(t\)

Syntax: \(-t\)
Description: Whether to format the enterprise id with the severity.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: set Insert severity into penultimate object of the extended enterprise id.
not set Do not format enterprise id with severity.
Default: not set

Resend Alarms

The smsAlarmRelay can be configured to listen for a request to resent all alarms above a certain alarm number. This is designed for use by an SNMP Manager that has been offline for a while and may have missed some alarm notifications.

To request a resend of alarms the relay application needs send an SNMP set-request using the format described in the variables.mib file.

The smsAlarmRelay will listen using the port number specified as the listenPort parameter in the snmp.cfg. The alarmRelay keeps an internal count of the highest alarm number sent. When a valid SNMP set-request is received, the alarmRelay will take note of the number in the message and send all alarms with an alarm Id greater than this number.

Failure

The smsAlarmDaemon on each alarm-managed node in the installation will by default generate a health-check alarm once per minute. These health check alarms will be relayed in the same fashion as all other alarms.

If these health check alarms are not received at the target destination, then the smsAlarmRelay may have failed, and should be investigated.

Continued on next page
The smsDbCleanup.sh writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/smsAlarmRelay.log

The following quick reference table summarises the information in the SMS User's Guide.

<table>
<thead>
<tr>
<th>Destination</th>
<th>Field Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNMP</td>
<td>the hostname of the target SNMP TRAPS recipient</td>
</tr>
<tr>
<td>FILE</td>
<td>the name of a file to which the daemon has write access</td>
</tr>
<tr>
<td>NFM</td>
<td>the host name of the NFM target.</td>
</tr>
<tr>
<td>Q3</td>
<td>the host name of the Q3 target.</td>
</tr>
<tr>
<td>SNMP</td>
<td>the host name of the SNMP target.</td>
</tr>
<tr>
<td>NORELAY</td>
<td>the field is empty, as the alarm is not forwarded to a target</td>
</tr>
</tbody>
</table>

Note: Setting the target to NORELAY will stop any other notification rules being actioned. Consequently, the NORELAY rules must be very specific. Otherwise an important alarm may accidentally be missed.
### smsConfigDaemon

**Purpose**

smsConfigDaemon exists on both the source node (example, USMS) as well as the target node (example, UAS). It takes an optional parameter (`-m`) which decides its action.

When run with the `-m`, it monitors for changes to the master XML file (example, esgConfigMaster.xml). If it finds changes made to the master config file, smsConfigDaemon will call smsSendConfig.sh.

If the `-m` parameter is missing, smsConfigDaemon monitors for changes to the derived eserv.config file (example, eserv.configderived) on the target node, and calls smsApplyConfig.sh if it finds changes to the file.

**Startup**

smsConfigDaemon is started by the script smsConfigDaemonScript. This process is driven by the system and is not intended to be changed by the user.

**Configuration**

For more information on the parameters used by smsConfigDaemon, see `smsConfigDaemonScript Configuration` (on page 103).

**Failure**

If the smsConfigDaemon fails, the secondary scripts, smsSendConfig.sh and smsApplyConfig.sh will fail to start and distribution of the updated configuration files is affected. Appropriate alarm messages are generated.

**Output**

The smsConfigDaemon and its sub-scripts write error messages to the system messages file, and also write additional output to:

/IN/service_packages/SMS/tmp/smsConfigDaemonMaster.log

or if they reside on the target node to:

/IN/service_packages/SMS/tmp/smsConfigDaemonClient.log
smsConfigDaemonScript

Purpose

smsConfigDaemonScript is responsible for starting the smsConfigDaemon process, depending on the supplied parameter (if applicable). It also loads the smsConfigVariables.sh, which sets the configurable variables and describes the directories used by smsConfigDaemon and its helper scripts.

For more information about the parameters used by smsConfigDaemon, see smsConfigDaemon (on page 102).

Startup

smsConfigDaemonScript is started by the system and is not intended to be changed by the user.

Configuration

smsConfigDaemonScript sets the configurable parameters for smsConfigDaemon and its helper scripts.

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smp_db_user</td>
<td>&lt;user/password&gt;</td>
<td>Oracle user/password for the SMF. Example: <code>smf/smf</code></td>
</tr>
<tr>
<td>scp_dbuser</td>
<td>&lt;user/password&gt;</td>
<td>Oracle user/password for the smsSignalConfigChange script should use for sqlplus. Example: <code>scp/scp</code></td>
</tr>
<tr>
<td>connect_string</td>
<td></td>
<td>Any extra connect parameters to be used by smsConfigDaemon</td>
</tr>
<tr>
<td>detection_period</td>
<td>10</td>
<td>Period (in seconds) after which smsConfigDaemon attempts to detect changes.</td>
</tr>
<tr>
<td>retry_period</td>
<td>60</td>
<td>Period (in seconds) after which smsConfigDaemon attempts to retry initiating sendConfig.</td>
</tr>
<tr>
<td>sleep_time</td>
<td>100</td>
<td>Period (in milliseconds) to wait inside smsConfigDaemon's main loop.</td>
</tr>
<tr>
<td>source_root</td>
<td>/IN/html/Configuration</td>
<td>Location where all the XML-driven config files and directories are stored. Not to be changed by the user.</td>
</tr>
<tr>
<td>master_xml_dir</td>
<td></td>
<td>Location of the master config.xml file. Not to be changed by the user.</td>
</tr>
<tr>
<td>master_config_file</td>
<td>esgConfigMas</td>
<td>Name of the master configuration xml file.</td>
</tr>
<tr>
<td>master_config_file_full_path</td>
<td></td>
<td>Full path of the master configuration file monitored by the SMP config daemon (derived from the master config xml file).</td>
</tr>
<tr>
<td>archive_xml_dir</td>
<td></td>
<td>Location where the master config.xml files are archived to prior to modification.</td>
</tr>
</tbody>
</table>

Continued on next page
**smsConfigDaemonScript**, Continued

### Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>derived_eserv_dir</td>
<td></td>
<td>Location of the derived eserv file.</td>
</tr>
<tr>
<td>pending_dir</td>
<td></td>
<td>Location where the config files from failed updates are held, pending a retry.</td>
</tr>
<tr>
<td>xml_convert_script</td>
<td></td>
<td>Location of the XML to eserv.config converter script.</td>
</tr>
<tr>
<td>target_root</td>
<td>/IN/service_packages/Configuration</td>
<td>Location of the USP nodes where the eserv.config file is sent.</td>
</tr>
<tr>
<td>target_eserv_config_dir</td>
<td></td>
<td>Location on the USP nodes where the eserv.config file is pushed out to.</td>
</tr>
<tr>
<td>derived_eserv</td>
<td>eserv.config.derived</td>
<td>Name of the eserv.config file sent to the target nodes.</td>
</tr>
<tr>
<td>derived_config_file</td>
<td></td>
<td>Full path of the derived config file monitored by the SCP config daemon</td>
</tr>
<tr>
<td>_full_path</td>
<td></td>
<td>(derived from the eserv.config file sent to the target nodes)</td>
</tr>
<tr>
<td>management_interface_host</td>
<td>localhost</td>
<td>Location of the management interface.</td>
</tr>
<tr>
<td>management_interface_port</td>
<td></td>
<td>Port is the management interface listening on.</td>
</tr>
</tbody>
</table>

**Note:** It is not recommended to change the values of these parameters. All necessary configuration is done at installation time by the configuration script; this section exists for information only. Please contact the Oracle support prior to attempting any modification to configuration data.

---

**Failure**

If smsConfigDaemonScript encounters problems, the smsConfigDaemon will fail to start and the updated eserv.config data will not be copied to the relevant platforms. Appropriate alarm messages are generated.

**Output**

The smsConfigDaemonScript writes error messages to the system messages file, and also writes additional output to the following when smsConfigDaemon has been started using the `--m` option:

```
/IN/service_packages/SMS/tmp/smsConfigDaemonMaster.log
```

If the `--m` option is not used, output will be written to:

```
/IN/service_packages/SMS/tmp/smsConfigDaemonClient.log
```
**smsCdrProcess.sh**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>smsCdrProcess.sh performs basic EDR processing and archiving. smsCdrProcess.sh runs the smsProcessCdr binary with specified command line parameters. One output EDR file is created for each input EDR file. smsCdrProcess.sh can also be configured to prevent processing the EDR. For more information about how smsProcessCdr processes EDRs, see <em>smsProcessCdr</em> (on page 179).</th>
</tr>
</thead>
<tbody>
<tr>
<td>EDR format</td>
<td>The format of the records in the EDR file are specific to the application which generates them. The most commonly used EDR format processed by this mechanism is the ACS &quot;Pipe Tag LF&quot; format, which uses TAG=VALUE pairs separated by the &quot;</td>
</tr>
<tr>
<td>Startup</td>
<td>This task is run in the crontab for smf_opr, by default at 1:00 am system clock time. It is scheduled as the following script: <code>/IN/service_packages/SMS/bin/smsCdrProcess.sh</code> The script runs the smsProcessCdr process with set parameters.</td>
</tr>
<tr>
<td>Configuration</td>
<td>The following command in the smsProcessCdr.sh prevents the EDR from being processed and copies it directly to the output directory. <strong>Example Command:</strong> <code>$BINDIR/smsProcessCdr -d $CDRDIR -D $OUTDIR -s $INSFX -s $OUTSFX</code> To process EDRs, use the following command instead: <strong>Example Command:</strong> <code>$BINDIR/smsProcessCdr -d $CDRDIR -D $OUTDIR -s $INSFX -s $OUTSFX</code></td>
</tr>
<tr>
<td>Failure</td>
<td>If the process is not running, EDR files will build up in the following directory: <code>/IN/service_packages/SMS/cdr/received</code> The filesystem usage will rise above standard operational levels.</td>
</tr>
<tr>
<td>Output</td>
<td>The smsCdrProcess.sh writes error messages to the system messages file, and also writes additional output to: <code>/IN/service_packages/SMS/tmp/smsCdrProcess.sh.log</code></td>
</tr>
</tbody>
</table>
smsDbCleanup.sh

**Purpose**
This task executes SQL statements to delete old data from the following tables.

<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>Data deleted</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMF_AUDIT</td>
<td>Audit trail of database changes.</td>
</tr>
<tr>
<td>SMF_STATISTICS</td>
<td>Application bulk usage counters.</td>
</tr>
<tr>
<td>SMF_TEMPLATE</td>
<td>Operator permissions templates.</td>
</tr>
<tr>
<td>SMF_ALARM_MESSAGE</td>
<td>System messages.</td>
</tr>
</tbody>
</table>

**Startup**
This task is run in the crontab for smf_oper, by default at 1:00 am system clock time. It is a shell script, specifically:

```
/IN/service_packages/SMS/bin/smsDbCleanup.sh
```

**Parameters**
The decision on when to delete data is determined according to various parameters configured in eserv.config. These values can be changed by the usual eserv.config editing method, subject to database sizing limitations and availability of space for additional historical data.

The default parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alarmAge</td>
<td>7</td>
<td>Clean up records older than this number of days.</td>
</tr>
<tr>
<td>alarmMax</td>
<td>100000</td>
<td>Clean up records once there are more than this number of records.</td>
</tr>
<tr>
<td>auditAge</td>
<td>7</td>
<td>Clean up records older than this number of days.</td>
</tr>
<tr>
<td>commit</td>
<td>100</td>
<td>Number of statistic records to delete before committing the deletions.</td>
</tr>
<tr>
<td>statsAge</td>
<td>30</td>
<td>Clean up records older than this number of days.</td>
</tr>
<tr>
<td>templateAge</td>
<td>1</td>
<td>Clean up templates not allocated to any operator older than this number of days.</td>
</tr>
<tr>
<td>unknownMax</td>
<td>5000</td>
<td>Maximum number of alarms to keep in table smf_alarm_unknown. After this value is reached, new additions cause oldest to be deleted from the table.</td>
</tr>
</tbody>
</table>

**Failure**
If the process is not running, old data will not be purged from the database. The database may reach maximum size, and inserts may fail.

**Output**
The smsDbCleanup.sh writes error messages to the system messages file, and also writes additional output to:

```
/IN/service_packages/SMS/tmp/smsDbCleanup.sh.log
```
smsLogCleaner

Purpose

The log cleaner looks for old/large files in the following directory:

/IN/service_packages/SMS/tmp

Any files found there, are moved into the archive subdirectory. smsLogCleaner then purges the files from the originating directory.

Startup

This task is run in the crontab for smf_oper. By default it runs at 30 minutes past each hour. It is run via the shell script:

/IN/service_packages/SMS/bin/smsLogCleanerStartup.sh

Parameters

The smsLogCleaner supports the following command-line options:

Usage:

smsLogCleaner -c <configuration file name> -d <days> -s <storage file name> [-h]

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c &lt;configuration file name&gt;</td>
<td>logjob.conf</td>
<td>The configuration file to use.</td>
</tr>
<tr>
<td>-d &lt;days&gt;</td>
<td>7</td>
<td>How often to clean archive in days.</td>
</tr>
<tr>
<td>-s &lt;storage file name&gt;</td>
<td>storage.txt</td>
<td>The storage file to use.</td>
</tr>
<tr>
<td>-h</td>
<td></td>
<td>Provides help information.</td>
</tr>
</tbody>
</table>

At installation time, the cronjob is configured to execute by default with the following command line parameters:

- c /IN/service_packages/SMS/etc/logjob.conf
- s /IN/service_packages/SMS/tmp/sms_storage.txt
- d 7

An operator may change these values, subject to disk storage availability and site-specific archiving policies.

Failure

If the process is not running, log files in the following directory will accumulate in size and age beyond the expected values.

/IN/service_packages/SMS/tmp

Output

The smsLogCleaner run by smf_oper writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/smsLogCleaner.log

Continued on next page
The logjob.conf configuration file has the following format:

```
log <file> age <hrs> size <size> arcdir <dir> logonce
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| log <file> | The full directory path and name of the file to be cleaned. You can include the '*' wildcard in the file name if required.  
Example:  
```
log /IN/service_packages/SMS/tmp/smsNamingServer.log
```
| age <hrs>   | Sets the minimum age in hours for the log file before it will be cleaned. You must set either this parameter or the size parameter. If both parameters are set, then the log file is cleaned if either condition is met.  
Example:  
```
age 100
```
| size <size> | Sets the minimum size for the logfile before it will be cleaned. You must set either this parameter or the age parameter. If both parameters are set, then the log file is cleaned if either condition is met.  
Examples:  
```
size 60K, or size 60M
```
| arcdir <dir>| The directory to use to store the old log file. If this parameter is not specified, then the log file is deleted.  
Example:  
```
arcdir /IN/service_packages/SMS/tmp/archive
```
| logonce    | Only specify this parameter if you just want to keep one archived version of the log file. |
smsMergeDaemon

Purpose
The smsMergeDaemon monitors the connections between the USMS and UASs via a heartbeat. The smsMergeDaemon will initiate a startMerge to resynchronise the USMS and UASs where:

- the infMaster on the disconnected UAS reports that it has received updates that would have normally gone to the USMS or it has an updateLoader or updateRequester pointing to it, and
- the heartbeat to the USMS and UAS have been stable for a period.

For more information about the startMerge process, see startMerge (on page 187).

Startup
This task is started by entry sms9 in the inittab, via the shell script:

/IN/service_packages/SMS/bin/smsMergeDaemonStartup.sh

Note: smsMergeDaemon is not used in a clustered install.

Parameters
The smsMergeDaemon accepts the following command line arguments.

Usage:

smsMergeDaemon -nodeid

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-nodeid</td>
<td>1000</td>
<td>The node number of the smsMergeDaemon.</td>
</tr>
</tbody>
</table>

The rest of the configuration details are taken from replication.def (on page 35). Relevant parameters include:

- HB PERIOD
- LONG TIMEOUT
- MAX_ROUNDTRIP 3
- MAX_CONNECTION_TIME 100000000
- MERGE_INTERVAL 600
- REP_PATH "/IN/service_packages/SMS/etc/replication.config"
- SMS_PORT 7
- TICK_TIME 1000

Failure
If the smsMergeDaemon's connection to the smsMaster is lost, it will exit.

Output
The smsMergeDaemon.sh writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/smsMergeDaemon.log
smsMaster

**Purpose**
The smsMaster is the central correlation point for the replication system.
smsMaster:
- sends notifications of updates to remote updateLoaders to be loaded into secondary databases.
- accepts update requests from remote systems that wish to change the master database (including the smsStatsDaemon, RequesterIF and smsAlarmDaemon).
- correlates full resynchronisation with remote databases, and communicates with inferior masters which can assume some smsMaster functions in the case of a platform or network failure.

**Startup**
This task is started by entry sms7 in the inittab, via the shell script:

/IN/service_packages/SMS/bin/smsMasterStartup.sh

**Configuration**
The smsMaster supports the following command-line options:

**Usage:**
smsMaster -maxpending <int>

The available parameters are:

- **-maxpending <int>**
  - **Syntax:** -maxpending <int>
  - **Description:** The size of pending request queue.
  - **Type:** Integer
  - **Optionality:** Optional (default used if not set).
  - **Default:** 10000

**Failure**
Remote replication nodes such as update loaders, updated requesters, inferior masters will generate alarms indicating connection failure to the smsMaster.

**Output**
The smsMaster writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/smsMaster.log
smsNamingServer

Introduction
The smsNamingServer listens for IORs being exported from CORBA processes, and stores them in the database in the IORS table owned by Oracle user SMF. It also serves requests to read IOR strings from the database.

This functionality is required to support processes that wish to store/retrieve IOR strings, but which do not have Oracle access to the SMF database instance, for security or licensing reasons.

Startup
In an unclustered installation, this task is started by entry sms2 in the inittab, via the shell script:

/IN/service_packages/SMS/bin/smsNamingServerStartup.sh

In a clustered installation this task is started by the cluster software, via the shell script:

/IN/service_packages/SMS/bin/smsNamingServerCluster.sh

Parameters
The smsNamingServer supports the following command-line options:

Usage:

```
smsNamingServer [-u <usr/pwd>] [-p <port>]
```

The available parameters are:

- **-u <usr/pwd>**
  
  Syntax: `-u <usr/pwd>`
  Description: The Oracle user and password pair.
  Type: String
  Optionality: Optional (default used if not set).
  Default: `/`

- **-p <port>**
  
  Syntax: `-p <port>`
  Description: The port number on which to listen for requests.
  Type: Integer
  Optionality: Optional (default used if not set).
  Default: 7362

Failure
If the smsNamingServer fails, then processes attempting to access the specified port will not be able to access the service, and should report an error indicating this.

Output
The smsNamingServer writes error messages to the system messages file, and also writes additional output to:

```
/IN/service_packages/SMS/tmp/smsNamingServer.log
```
smsReportsDaemon

### Purpose

The smsReportsDaemon is a CORBA server process which generates reports on demand, as requested by the Reports function of the SMS Java screens. The Java screens will poll until the reports daemon indicates successful completion, and returns the output filename.

The smsReportsDaemon writes the output into a directory on the USMS which is available to the Java screens via the web-server. The Java screens will fetch the report text, and display that to the operator.

### Startup

In an unclustered installation, this task is started by entry sms3 in the initdb, via the shell script:

```
/IN/service_packages/SMS/bin/smsReportsDaemonStartup.sh
```

In a clustered installation this task is started by the cluster software, via the shell script:

```
/IN/service_packages/SMS/bin/smsReportsDaemonCluster.sh
```

### Parameters

The smsReportsDaemon supports the following command-line options:

**Usage:**

```
smsReportsDaemon -h <host> -p <port> -i <dir> -o <dir> -u <user/password> -t <host> -s <port> -z <timezone>
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-h &lt;host&gt;</td>
<td>value returned by gethostname</td>
<td>smsNamingServer hostname. Allowed value type: ASCII String</td>
</tr>
<tr>
<td>-p &lt;port&gt;</td>
<td>7362</td>
<td>smsNamingServer port number. Allowed value type: Number</td>
</tr>
<tr>
<td>-i &lt;dir&gt;</td>
<td>/IN/service_packages/SMS/input</td>
<td>Report scripts/binaries input directory. Allowed value type: ASCII String</td>
</tr>
<tr>
<td>-o &lt;dir&gt;</td>
<td>/IN/service_packages/SMS/output</td>
<td>Generated report output directory. Allowed value type: ASCII String</td>
</tr>
<tr>
<td>-u &lt;user/password&gt;</td>
<td></td>
<td>Oracle SMF database username and password. Allowed value type: ASCII String</td>
</tr>
<tr>
<td>-t &lt;host&gt;</td>
<td>default determined by CORBA</td>
<td>CORBA listener ILU transport hostname.</td>
</tr>
<tr>
<td>-s &lt;port&gt;</td>
<td>default determined by CORBA</td>
<td>CORBA listener ILU transport port number.</td>
</tr>
<tr>
<td>-z &lt;timezone&gt;</td>
<td></td>
<td>Timezone in which the smsReportsDaemon SQL queries are run generating the report output.</td>
</tr>
</tbody>
</table>

*Continued on next page*
If the smsReportsDaemon fails, then user attempts to generate reports will do nothing, and will display a message box indicating failure after approximately five minutes.

The smsReportsDaemon writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/smsReportsDaemon.log

The smsReportsDaemon generates report output to subdirectories of the specified output directory, by default:

/IN/service_packages/SMS/output

The subdirectory depends on the application and category defined for the report.

/IN/service_packages/SMS/output/<Application>/<Category>

The filename for the report output is:

YYMMDDHHmmss.<9-random-characters>.txt

The use of the Reports screens is described in the SMS User's Guide.

The smsReportsDaemon generates on-demand reports. The smsReportsScheduler is responsible for generating scheduled reports.

The smsReportsCleaner is responsible for cleaning up the output from old reports.

Reports may be defined using SQL commands, by shell scripts, or by compiled executable programs. Additional reports may be created and made available for on-demand and scheduled generation as a post-installation manual function. The mechanism for doing this is described (elsewhere in this document).

At startup, the smsReportsDaemon publishes its IOR string via the smsNamingServer. If not specified, the IP port number on which the CORBA service is provided will be determined by the CORBA framework. In most installations, a firewall is used to protect the USMS host, and hence the CORBA service port must be fixed. Use the "-s" parameter for this purpose.
The smsReportScheduler monitors the database table SMF_REPORT_SCHEDULE for entries inserted via the SMS Java screens. The report scheduler will sleep until the next report is due to be executed. The output of the report will be optionally copied to a specified directory, spooled to a specified printer, or sent to a specified email address. For more information about how to schedule reports which will be performed periodically and how to configure the report destination, see the SMS User's Guide.

In an unclustered installation, this task is started by entry sms4 in the inittab, via the shell script:

/IN/service_packages/SMS/bin/smsReportSchedulerStartup.sh

In a clustered installation this task is started by the cluster software, via the shell script:

/IN/service_packages/SMS/bin/smsReportSchedulerCluster.sh

The smsReportScheduler supports the following command-line options:

Usage:
smsReportScheduler [-i <dir>] [-o <dir>] [-u <usr/pwd>] [-v]

The available parameters are:

-i <dir>

Syntax: -i <dir>
Description: The input directory for report generation scripts/binaries dir.
Type: String
Optionality: Optional (default used if not set).
Default: /IN/service_packages/SMS/input

-o <dir>

Syntax: -o <dir>
Description: The output directory for report generation.
Type: String
Optionality: Optional (default used if not set).
Default: /IN/service_packages/SMS/output
Notes: Report may provide a default output directory which overrides smsReportDaemon's default.

-u <usr/pwd>

Syntax: -u <usr/pwd>
Description: The userid and password for oracle login string.
Type: String
Optionality: Optional (default used if not set).
Default: /
smsReportScheduler, Continued

Parameters (continued)

\[-v\]

- Syntax: \(-v\)
- Description: What level of information to output.
- Type: Boolean
- Optionality: Optional (default used if not set).
- Allowed:
  - set: Print additional information.
  - not set: Only print the standard level of information.
- Default: not set

Failure
In the case of failure, the scheduled report will not appear at the specified destination, or may contain incorrect or missing output.

Output
The smsReportScheduler writes error messages to the system messages file, and also writes additional output to:

\(/IN/service_packages/SMS/tmp/smsReportScheduler.log\)

Unix utilities
The table below lists the Unix utilities required for scheduling.

<table>
<thead>
<tr>
<th>Unix Binary Required</th>
<th>Description</th>
<th>Location Expected</th>
</tr>
</thead>
<tbody>
<tr>
<td>mailto</td>
<td>E-mail agent used by report generation component to send emails.</td>
<td>/usr/bin/mailto</td>
</tr>
</tbody>
</table>
| sendmail (or equivalent delivery agent) | E-mail delivery agent.  
| lpr                  | Printing utility.                                                           | /usr/ucb/lpr      |

Note: E-mail sending/receiving/delivery agent requires all local e-mail user names to be under 13 characters. For local e-mail user names longer than 13 characters, mailto and sendmail will not function properly.
### smsReportCleanupStartup.sh

<table>
<thead>
<tr>
<th>Purpose</th>
<th>The Reports cleaner looks for output from ad-hoc and scheduled reports generated by the smsReportsDaemon and the smsReportScheduler. It deletes files that are older than a specified age.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup</td>
<td>This task is run in the crontab for smf_oper. By default it runs at 2:00 am system time. It is scheduled as the following script:</td>
</tr>
<tr>
<td></td>
<td>/IN/service_packages/SMS/bin/smsReportsCleanerStartup.sh</td>
</tr>
<tr>
<td>Parameters</td>
<td>The command inside the script contains a command line parameter specifying a cleanup time of 7 days. Report output files older than this age are deleted. An operator may change this value, subject to disk storage availability and site-specific archiving policies.</td>
</tr>
<tr>
<td>Failure</td>
<td>If the process is not running, reports files in the following directory will accumulate in size and age beyond the expected values.</td>
</tr>
<tr>
<td></td>
<td>/IN/service_packages/SMS/output</td>
</tr>
<tr>
<td>Output</td>
<td>The smsReportsCleaner run by smf_oper writes error messages to the system messages file, and also writes additional output to:</td>
</tr>
<tr>
<td></td>
<td>/IN/service_packages/SMS/tmp/smsReportsCleanerStartup.sh.log</td>
</tr>
</tbody>
</table>
### smsStatsDaemon

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsStatsDaemon can be run as a background process on an USMS, UAS and also on other IPs such as Billing Engines. For more configuration information about the process and it's parameters, see <em>smsStatsDaemon</em> (on page 141).</td>
</tr>
</tbody>
</table>
### smsStatsThreshold

**Purpose**
The `smsStatsThreshold` polls the database for updates to the SMF_STATISTICS table. It compares the values against threshold rules defined in the SMF_STATISTICS_RULE table, and raises an alarm if the threshold is exceeded. It inserts the alarm into the SMF_ALARM_MESSAGE table in the SMF database.

For more information about how to define new statistics threshold rules, see the SMS User's Guide. New threshold rules are automatically recognised by the program.

**Startup**
In an unclustered installation, this task is started by entry sms6 in the inittab, via the shell script:

```
/IN/service_packages/SMS/bin/smsStatsThresholdStartup.sh
```

In a clustered installation, this task is started by the cluster software, via the shell script:

```
/IN/service_packages/SMS/bin/smsStatsThresholdCluster.sh
```

**Parameters**
The `smsStatsThreshold` supports the following command-line options:

**Usage:**

```
smsStatsThreshold -u <usr/pwd> -s <secs>
```

The available parameters are:

- `-u <usr/pwd>`
  - **Syntax:** `-u <usr/pwd>`
  - **Description:** The Oracle user and password pair.
  - **Type:** String
  - **Optionality:** Optional (default used if not set).
  - **Default:** `/`

- `-s <secs>`
  - **Syntax:** `-s <secs>`
  - **Description:** The number of seconds to sleep between database checks.
  - **Type:** Integer
  - **Optionality:** Optional (default used if not set).
  - **Default:** 60

**Failure**
Alarm messages derived from statistics values will not appear in the alarm system.

**Output**
The `smsStatsThreshold` writes error messages to the system messages file, and also writes additional output to:

```
/IN/service_packages/SMS/tmp/smsStatsThreshold.log
```
**smsSendConfig.sh**

<table>
<thead>
<tr>
<th>Purpose</th>
<th>smsSendConfig.sh resides on the source node (for example, an USMS) and performs the following functions:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• archives current master XML file</td>
</tr>
<tr>
<td></td>
<td>• stores audit information</td>
</tr>
<tr>
<td></td>
<td>• sets link to current archived file</td>
</tr>
<tr>
<td></td>
<td>• converts XML format to derived eserv.config using the script cmnConfigXmlConvert.sh, and</td>
</tr>
<tr>
<td></td>
<td>• sends derived eserv.config file to target node using scp.</td>
</tr>
</tbody>
</table>

| Startup | smsSendConfig.sh is started by smsConfigDaemon (using the -m parameter). It is driven by the system and is not intended to be changed by the user. |

| Configuration | For more information on the parameters used by smsSendConfig.sh, see smsConfigDaemonScript Configuration (on page 103). |

| Failure | If smsSendConfig.sh fails, deployment process for the eserv.config on the source node will fail. Consequently, no updates will be sent to the target node. Appropriate alarm messages are generated. |

<table>
<thead>
<tr>
<th>Output</th>
<th>The smsSendConfig.sh and its sub-scripts write error messages to the system messages file, and also write additional output to:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/IN/service_packages/SMS/tmp/smsConfigDaemonMaster.log</td>
</tr>
</tbody>
</table>
smsTaskAgent

Purpose

The smsTaskAgent is a CORBA server which performs various utility functions as requested by the SMS Java screens. The tasks performed by the smsTaskAgent are:

- Create/Transfer new replication.config file.
- Change SMS operator Oracle password.
- Perform data consistency checks with remote nodes.

CORBA service port

At startup, the smsTaskAgent publishes its IOR string via the smsNamingServer. If not specified, the IP port number on which the CORBA service is provided will be determined by the CORBA framework. In most installations, a firewall is used to protect the USMS host, and hence the CORBA service port must be fixed. Use the "-s" parameter for this purpose.

Startup

In an unclustered installation, this task is started by entry sms8 in the inittab, via the shell script:

```
/IN/service_packages/SMS/bin/smsTaskAgentStartup.sh
```

In a clustered installation, this task is started by the cluster software, via the shell script:

```
/IN/service_packages/SMS/bin/smsTaskAgentCluster.sh
```

Parameters

The smsTaskAgent supports the following command-line options:

**Usage:**

```
smsTaskAgent [-c] [-i <ior host>] [-p <ior port>] [-u <usr/pwd>] [-t <trans host>] [-s <trans port>] [-w <secs>] [-g]
```

The available parameters are:

- **-c**
  - Syntax: `-c`
  - Description: Use secure shell and secure copy (ssh and scp).
  - Type: Boolean
  - Optionality: Optional (default used if not set).
  - Default: Use standard connection and copy.

- **-i <host>**
  - Syntax: `-i <host>`
  - Description: The IOR Listener host to connection to.
  - Type: String
  - Optionality: Optional (default used if not set).
  - Default: localhost
  - Example: `-i produsms`

Continued on next page
smsTaskAgent, Continued

Parameters (continued)

-\texttt{p <port>}
  \begin{itemize}
  \item Syntax: \texttt{-p <port>}
  \item Description: The port on the IOR Listener host to connect to.
  \item Type: Integer
  \item Optionality: Optional (default used if not set).
  \item Default: 5556
  \item Example: \texttt{-p 13579}
  \end{itemize}

-\texttt{u <usr/pwd>}
  \begin{itemize}
  \item Syntax: \texttt{-u <usr/pwd>}
  \item Description: The userid and password to use to log into Oracle.
  \item Type: String
  \item Optionality: Optional (default used if not set).
  \item Default: /
  \end{itemize}

-\texttt{t <host>}
  \begin{itemize}
  \item Syntax: \texttt{-t <host>}
  \item Description: The ILU transport host to connect to.
  \item Type: String
  \item Optionality: Optional (default used if not set).
  \item Default: NULL
  \end{itemize}

-\texttt{s <port>}
  \begin{itemize}
  \item Syntax: \texttt{-s <port>}
  \item Description: The port on the ILU transport host to connect to.
  \item Type: Integer
  \item Optionality: Optional (default used if not set).
  \item Default: 0
  \end{itemize}

-\texttt{w <secs>}
  \begin{itemize}
  \item Syntax: \texttt{-w <secs>}
  \item Description: The number of seconds smsTaskAgent waits for a consistency check update before timing out and abandoning a consistency check.
  \item Type: Integer
  \item Optionality: Optional (default used if not set).
  \item Default: 20
  \item Example: \texttt{-w 120}
  \end{itemize}

\textit{Continued on next page}
**smsTaskAgent**, Continued

### Parameters (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Description</th>
<th>Type</th>
<th>Optionality</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-g</td>
<td>-g</td>
<td>Use signal handler to detect whether child process</td>
<td>Boolean</td>
<td>Optional (default</td>
<td>Use waitpid.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(consistency check) has finished.</td>
<td></td>
<td>used if not set)</td>
<td></td>
</tr>
</tbody>
</table>

**Failure**

If the smsTaskAgent fails, then user attempts to perform the tasks served by this smsTaskAgent will do nothing, and will display a message box indicating failure after approximately five minutes.

**Output**

The smsTaskAgent writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/smsTaskAgent.log
smsTrigDaemon

**Purpose**

smsTrigDaemon manages control plan execution requests. It runs on the USMS platform.

smsTrigDaemon accepts control plan execution requests from either a remote PI client or the Java management screens. It forwards requests to ACS through the xmlTcapInterface on the UAS platform. An indication of whether or not the requests were successful passes back from the ACS to the initiating client.

**Startup**

In an unclustered installation, this task is started by entry sm11 in the inittab, via the shell script:

```
/IN/service_packages/SMS/bin/smsTrigDaemonStartup.sh
```

In a clustered environment this task is started by the binary startSmsTrigDaemon which is located in:

```
/opt/ESERVSmsTrigDaemon/util/startSmsTrigDaemon
```

Note: startSmsTrigDaemon must be manually run as root in a clustered environment. smsTrigDaemon is then added as a cluster resource.

**Location**

This binary is located on USMSs.

**Parameters**

smsTrigDaemon is configured by the following parameters from the triggering section of eserv.config:

**Usage:**

```
triggering = {
  oracleLogin = "<userName>/<password>"
  useORB = <true|false>
  listenPort = <portNumber>
  useFIFO = <true|false>
  extraFIFO = [ 
    "<1stPath>"
    "<2ndPath>"
    ...
    ...
    "<nthPath>"
  ]
  scps = [ 
    "<1stHostAddress>:<1stPortNumber>"
    "<2ndHostAddress>:<2ndPortNumber>"
    ...
    ...
    "<nthHostAddress>:<nthPortNumber>"
  ]
}
```

Available parameters are:

Continued on next page
**smsTrigDaemon, Continued**

### Parameters (continued)

**oracleLogin**
- **Syntax:** `oracleLogin = "<usr>/<pwd>"`
- **Description:** The Oracle user name and password that smsTrigDaemon uses when connecting to the database.
- **Type:** String
- **Optionality:** Mandatory
- **Default:** `/`

**useORB**
- **Syntax:** `useORB = <true|false>`
- **Description:** Whether smsTrigDaemon accepts incoming CORBA requests.
- **Type:** Boolean
- **Optionality:** Mandatory
- **Allowed:**
  - `true`: smsTrigDaemon accepts incoming CORBA requests.
  - `false`: smsTrigDaemon refuses incoming CORBA requests.
- **Default:** `false`

**listenPort**
- **Syntax:** `listenPort = <port>`
- **Description:** The IP port on which smsTrigDaemon listens for CORBA requests.
- **Type:** Integer
- **Optionality:** Mandatory
- **Allowed:** 0 - 65535
- **Default:** 0
- **Notes:** If set to 0, any available port is used.

*Continued on next page*
smsTrigDaemon, Continued

Parameters (continued)

-useFIFO

Syntax: useFIFO = <true|false>

Description: Whether smsTrigDaemon accepts FIFO transport layer incoming requests.

Type: Boolean

Optionality: Mandatory

Allowed: true smsTrigDaemon accepts FIFO transport layer incoming requests

false smsTrigDaemon refuses FIFO transport layer incoming requests

Default: false

Notes: If set to false, the extraFIFO (on page 125) parameter array is ignored.

If set to false and useORB (on page 124) is also set to false, smsTrigDaemon will do nothing.

extraFIFO

Syntax: extraFIFO = [
    "<dir>"
    ...
]

Description: The paths that smsTrigDaemon should create for extra FIFOs.

Type: Parameter array

Optionality: Optional

Default: Empty set

Notes: If useFIFO (on page 125) is set to false, this parameter array is ignored.

Example: extraFIFO = [
    "/IN/service_packages/SMS/tmp/trg-req-3005"
    "/IN/service_packages/SMS/tmp/trg-req-3006"
]
smsTrigDaemon, Continued

Parameters (continued)

**scps**

**Syntax:**
```
scps = [
  "<ip>:<port>"
  ...
]
```

**Description:**
The IP address and port number for each UAS that smsTrigDaemon connects to.

**Type:**
Array

**Optionality:**
The scps parameter array is optional.
In any row of the array, the :<port> part is optional.

**Allowed:**
- `<ip>` Any IP address in either dotted decimal or symbolic name format.
- `<port>` 0 - 65535

**Default:**
Empty set

**Notes:**
An example of an Internet protocol address in dotted-decimal format is: 192.168.1.56.
An example of an address in symbolic name format is: primary_smc.

**Example:**
```
scps = [
  "192.168.1.5:3005"
  "192.168.1.6"
  "primary_smc"
  "secondary_smc:3006"
]
```

---

**Failure**

If smsTrigDaemon fails, then interaction with the BPL requests from the Java screens and the PI will fail.

**Output**

smsTrigDaemon writes error messages to the system messages file, and also writes additional output to:

```
/IN/service_packages/SMS/tmp/smsTrigDaemon.log
```

Continued on next page
After smsTrigDaemon receives a control plan execution request, it follows a three-stage process.

### Stage Description

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | smsTrigDaemon attempts to connect to one of the UASs in the following way.  
        - If a connection to an UAS is established and not in use, that connection is used. smsTrigDaemon maintains a list of currently-open connections.  
        - If a connection to an UAS is not established, smsTrigDaemon attempts to open one. UASs are identified by the `scps` configuration parameter. smsTrigDaemon polls this list until it finds an available connection. If the connection fails, the next UAS in the list is tried. |
| 2     | smsTrigDaemon connects to the UAS using either port 3072 or another port specified by the `scps` parameter.  
        - If a connection to an UAS is established and not in use, that connection is used. smsTrigDaemon maintains a list of currently-open connections.  
        - If a connection to an UAS is not established, smsTrigDaemon attempts to open one. UASs are identified by the `scps` configuration parameter. smsTrigDaemon polls this list until it finds an available connection. If the connection fails, the next UAS in the list is tried. |
| 3     | smsTrigDaemon sends an XML message via an HTTP "POST / HTTP/1.1" request. The syntax of the message is:  
        `<control-plan-exec>`  
        `<control-plan><name of control plan></control-plan>`  
        `<service-handle><service handle></service-handle>`  
        `<cgpn><calling party></cgpn>`  
        `<cdpn><called party></cdpn>`  
        `<ext id="400"><host></ext>`  
        `<ext id="401"><user></ext>`  
        `<ext id="402"><more extensions></ext>`  
        ...  
        `</control-plan-exec>`  
        At least two extension parameters, id="400" and id="401", will be present. These represent the client's host and user names. Optional additional extension parameters can be included with labels id="402", id="403", etc. |

### Data consistency check

It is possible for a race condition to occur after the **Save & Execute** button has been pressed in a Java management screen. The race condition can exist between SMS's replication system and execution on the UASs of an smsTrigDaemon request.

To avoid this possibility, a data consistency check is carried out on the current subscriber before proceeding with the request.

*Continued on next page*
Data consistency check (continued)

Because it is not possible to know in advance which UAS will be selected by smsTrigDaemon, a data consistency check is performed on all replicated UASs. A decision to carry on with the request is only made after the check has been completed.

The following steps describe the consistency check process and the criteria used to determine whether the execution will be allowed.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Java management screen sends to smsTaskAgent a request for a consistency check on the current subscriber.</td>
</tr>
<tr>
<td>2</td>
<td>The Java management screen waits until a check report is received from smsTaskAgent. The report is in the form of an HTML file.</td>
</tr>
<tr>
<td>3</td>
<td>The Java management screen extracts relevant information from the report, including:</td>
</tr>
<tr>
<td></td>
<td>• The number of nodes checked.</td>
</tr>
<tr>
<td></td>
<td>• The number of nodes that failed the check.</td>
</tr>
<tr>
<td></td>
<td>• The number of nodes that replied to the check request.</td>
</tr>
<tr>
<td></td>
<td>• The number of nodes that reported inconsistent data.</td>
</tr>
<tr>
<td>4</td>
<td>The consistency check:</td>
</tr>
<tr>
<td></td>
<td>• succeeds, if:</td>
</tr>
<tr>
<td></td>
<td>▪ at least one node replied without error, and</td>
</tr>
<tr>
<td></td>
<td>▪ no node reported inconsistent data.</td>
</tr>
<tr>
<td></td>
<td>• fails, if:</td>
</tr>
<tr>
<td></td>
<td>▪ no nodes replied without error, or</td>
</tr>
<tr>
<td></td>
<td>▪ one or more nodes reported inconsistent data.</td>
</tr>
<tr>
<td>5</td>
<td>If the check succeeds, the request proceeds.</td>
</tr>
<tr>
<td>6</td>
<td>If the request fails, steps 1 through 4 are repeated. After three fails, the request is cancelled and the user informed with an SMS message dialogue.</td>
</tr>
</tbody>
</table>
## Overview

**Introduction**

This chapter provides a description of the programs or executables used by the System as background processes on an UAS.

**Executables are located in the /IN/service_packages/SMS/bin directory.**

Some executables have accompanying scripts that run the executables after performing certain cleanup functions. All scripts should be located in the same directory as the executable.

**Important:** It is a pre-requisite for managing these core service functions that the operator is familiar with the basics of Unix process scheduling and management. Specifically, the following Unix commands:

- `init` (and `inittab`)
- `cron` (and `crontab`)
- `ps`
- `kill`

**In this chapter**

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>smsApplyConfig.sh</td>
<td>130</td>
</tr>
<tr>
<td>cmnPushFiles</td>
<td>131</td>
</tr>
<tr>
<td>infMaster</td>
<td>136</td>
</tr>
<tr>
<td>smsAlarmDaemon</td>
<td>137</td>
</tr>
<tr>
<td>smsLogCleaner</td>
<td>139</td>
</tr>
<tr>
<td>smsStatsDaemon</td>
<td>141</td>
</tr>
<tr>
<td>updateLoader</td>
<td>150</td>
</tr>
</tbody>
</table>
### smsApplyConfig.sh

**Purpose**

smsApplyConfig.sh resides on the target node, example UAS.

It performs the following functions:

- Backup of the live eserv.config currently used in production,
- Merges changes from derived file into live eserv.config using `smsConfigSurgeon` (on page 55).
- Signals changes using `smsSignalConfigChanges.sh`.

If a SIGHUP is required, `smsSignalConfigChanges.sh` will in turn call `smsSendSighup.sh` (which will run with root permissions).

**Startup**

smsApplyConfig.sh is started by `smsConfigDaemon` (without the `-m` parameter). It is driven by the system and is not intended to be changed by the user.

**Configuration**

For more information on the parameters used by smsApplyConfig.sh, see `smsConfigDaemonScript Configuration` (on page 103).

**Failure**

If smsApplyConfig.sh fails, the deployment process for eserv.config on the target node will fail. Appropriate alarm messages will be generated.

**Output**

The smsApplyConfig.sh and its sub-scripts write error messages to the system messages file, and also write additional output to:

`/IN/service_packages/SMS/tmp/smsConfigDaemonClient.log`
cmnPushFiles

Purpose
cmnPushFiles transfers files to specific directories on the USMS from UASs and UBEs. The files transferred include:

- EDRs, and
- PIN logs.

Note: Other Oracle applications also use their own instances of this process.

Startup
This task is started by entry scp1 in the inittab, via the shell script:
/IN/service_packages/SMS/bin/cmnPushFilesStartup.sh

Configuration
cmnPushFiles accepts the following command-line options:

Usage:

- d <dir>
  Syntax:   -d <dir>
  Description: The destination directory for files on remote machine.
  Type:     String
  Optionality: Optional (default used if not set).
  Allowed:  Path must start with '/' or the -r option must also be used.
  Cannot be the same as -f <dir> (on page 133).
  Default:  .
  Notes:    An example of a destination directory is the directory on a UAS where cmnPushFiles looks for the files to be sent to the USMS.

- P <dir>
  Syntax:      -P <dir>
  Description: The file prefix to match on.
  Type:        String

- S <sufx>
  Syntax:      -S <sufx>
  Description: The file suffix.
  Type:        String

Continued on next page
cmnPushFiles, Continued

Configuration (continued)

-\texttt{r} \texttt{<pref>}

Syntax: \texttt{-r \textless pref\textgreater}

Description: The remote directory prefix.

Type: String

Optionality: Optional (default used if not set).

Default: null

Notes: Required if \texttt{-d \textless dir\textgreater} (on page 131) is a relative directory.

-\texttt{h} \texttt{<host>}

Syntax: \texttt{-h \textless host\textgreater}

Description: The hostname of the remote machine.

Type: String

Optionality: Mandatory

Default: null

Notes: If set, a hostname must be specified.

-\texttt{p} \texttt{<port>}

Syntax: \texttt{-p \textless port\textgreater}

Description: The port number on the remote machine on which cmnReceiveFiles will listen for receiving files.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: port Port to connect to.

\texttt{-1} Use stdin and stdout.

Default: 2027

-\texttt{s} \texttt{<secs>}

Syntax: \texttt{-s \textless secs\textgreater}

Description: The number of seconds for the sleep period.

Type: Integer

Optionality: Optional (default used if not set).

Default: 15

-\texttt{t} \texttt{<bits>}

Syntax: \texttt{-t \textless bits\textgreater}

Description: The number of bits per second to start throttling at.

Type: Integer

Optionality: Optional (default used if not set).

Default: 0 (no throttling)
cmnPushFiles, Continued

Configuration (continued)

-\( w \ <\text{secs}> \)
  Syntax: \( -w \ <\text{secs}> \)
  Description: The number of seconds to wait for success.
  Type: Integer
  Optionality: Optional (default used if not set).
  Default: 30

-\( x \)
  Syntax: \( -x \)
  Description: Whether to use hostname-prefixing on remote filenames.
  Type: Boolean
  Optionality: Optional (default used if not set).
  Allowed: set Don't use prefixing.
  (false)
  not set Use prefixing.
  (true)
  Default: true

-\( o \ <\text{dir}> \)
  Syntax: \( -o \ <\text{dir}> \)
  Description: The directory to transfer sent files to.
  Type: String
  Optionality: Optional (default used if not set).
  Allowed: directory The directory to store transferred files in.
  null Delete the transferred files, do not store them.
  Default: null (file deleted)

-\( f \ <\text{dir}> \)
  Syntax: \( -f \ <\text{dir}> \)
  Description: The retry directory.
  Type: String
  Optionality: Optional (default used if not set).
  Allowed: Cannot be the same as \( -d \ <\text{dir}> \) (on page 131).
  Default: null (no retry directory)

Continued on next page
cmnPushFiles, Continued

Configuration (continued)

- F
  Syntax: parameter = <>
  Description: Use fuser to not move files in use.
  Type: Boolean
  Optionality: Optional (default used if not set).
  Allowed: set (true) Use fuser.
  not set (false) Don't use fuser.
  Default: false

- a <days>
  Syntax: -a <days>
  Description: The number of days old a transferred file can be before it is deleted.
  Type: Integer
  Optionality: Optional (default used if not set).
  Allowed: positive integer
  -1 never delete files.
  Default: -1
  Notes: This parameter only relevant when -o <dir> (on page 133) option is specified.

- e
  Syntax: -e
  Description: Which mode to run in.
  Type: Boolean
  Optionality: Optional (default used if not set).
  Allowed: set Run in non-daemon mode. Execute file transfer only once, then exit.
  not set Run in daemon mode.
  Default: not set

- R <secs>
  Syntax: -R <secs>
  Description: The number of seconds before Initial retry period starts.
  Type: Integer
  Optionality: Optional (default used if not set).
  Default: 15
**cmnPushFiles**, Continued

### Configuration (continued)

- **-M <secs>**
  - **Syntax:** `-M <secs>`
  - **Description:** The maximum number of seconds for the retry period to continue.
  - **Type:** Integer
  - **Optionality:** Optional (default used if not set).
  - **Default:** 900

- **-C <secs>**
  - **Syntax:** `-C <secs>`
  - **Description:** The number of seconds for the cleanup period.
  - **Type:** Integer
  - **Optionality:** Optional (default used if not set).
  - **Default:** 1800

- **-T**
  - **Syntax:** `-T`
  - **Description:** Whether or not to move recursively.
  - **Type:** Boolean
  - **Optionality:** Optional (default used if not set).
  - **Allowed:**
    - set: Tree move: recursive into subdirectories.
    - not set
  - **Default:** true

**Example**

This text shows an example of the command line options for cmnPushFiles.

```
$ cmnPushFiles -d /IN/service_packages/SMS/cdr/closed -f /IN/service_packages/SMS/cdr/retry -r /IN/service_packages/SMS/cdr/received -h prodsmpl.telcoexample.com -s 10 -p 2028 -S cdr -w 20
```

**Failure**

If cmnPushFiles fails, EDRs will accumulate in:

```
/IN/service_packages/SMS/cdr/current/
```

cmnPushFiles will send error messages to the syslog and the cmnPushFiles log.

**Output**

The cmnPushFiles writes error messages to the system messages file, and also writes additional output to this default location:

```
/IN/service_packages/SMS/tmp/cmnPushFiles.log
```
infMaster

**Purpose**
The infMaster provides resilience for replication in case the smsMaster fails. For more information, see Inferior Master.

The infMaster is only used in the unclustered configuration.

**Note:** The infMaster does not replicate Alarms or Statistics.

**Startup**
This task is started by entry scp2 in the inittab, via the shell script:

```
/IN/service_packages/SMS/bin/infMasterStartup.sh
```

**Parameters**
The infMaster supports the following command-line options:

**Usage:**
```
infMaster [-maxpending <number>]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-maxpending</td>
<td>10000</td>
<td>This sets the maximum number of pending updates which will be queued in the infMaster's memory.</td>
</tr>
</tbody>
</table>

**Failure**
If the infMaster fails, no functionality will be affected unless the infMaster would normally be required to operate as the Superior Master (that is, the smsMaster and all other infMasters with higher node numbers were unavailable). In this case, replication will not work.

The infMaster will send error messages to syslog and infMaster.log.

**Output**
The infMaster writes error messages to the system messages file, and also writes additional output to:

```
/IN/service_packages/SMS/tmp/infMaster.log
```
smsAlarmDaemon

**Purpose**

The smsAlarmDaemon runs on all alarm-managed nodes in the SMS system, including the USMS nodes. The role of the smsAlarmDaemon is to gather alarms from the following sources:

- System error log (/var/adm/syslog.log or /var/log/syslog)
- Oracle Standard DB error log
- Sigtran SUA logs (/IN/service_packages/SLEE/stats) [If installed]

On the USMSs, the resultant error messages are written directly into the SMF_ALARM_MESSAGE table in the SMF. When run on other nodes, replication is used to update the SMF_ALARM_MESSAGE table.

**Startup**

This task is started by entry scp4 in the inittab, via the shell script:

```
/IN/service_packages/SMS/bin/smsAlarmDaemonScpStartup.sh
```

**Configuration**

The smsAlarmDaemon accepts the following command line arguments.

**Usage:**

```
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a &lt;path&gt;</td>
<td>Null</td>
<td>Location of the Oracle Alert Log. By default no oracle alert log will be generated.</td>
</tr>
<tr>
<td>-c</td>
<td>1</td>
<td>Commit Rate - number of inserts before committing to the database.</td>
</tr>
<tr>
<td>-d</td>
<td>Sort messages</td>
<td>Disable sorting of messages by severity in buffer. Specifically, messages are kept in the buffer and subsequently written into the SMF database, in the same sequence in which they are received.</td>
</tr>
<tr>
<td>-f</td>
<td>No filtering</td>
<td>Filtering - delete duplicate alarms and increase the alarm count.</td>
</tr>
<tr>
<td>-g</td>
<td>Uses local time</td>
<td>GMT timezone - use GMT instead of local time.</td>
</tr>
<tr>
<td>-h &lt;secs&gt;</td>
<td>60</td>
<td>Heartbeat message. Will be forced to be greater or equal to time period (seconds).</td>
</tr>
<tr>
<td>-i</td>
<td>Use fuzzy matching</td>
<td>Filtering type - use exact matching (rather than fuzzy matching). Indicates that duplicate matches should be performed on text only (that is, excluding digits). Note: Only valid when used in conjunction with -f.</td>
</tr>
<tr>
<td>-l &lt;secs&gt;</td>
<td>2</td>
<td>Filter Period - duration between linked-list checks (seconds).</td>
</tr>
<tr>
<td>-n &lt;number&gt;</td>
<td>5</td>
<td>Filter Number - number of alarm messages allowed within the time period. Allowed values: integers</td>
</tr>
</tbody>
</table>

Continued on next page
smsAlarmDaemon, Continued

Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-m &lt;max&gt;</td>
<td>1000</td>
<td>Maximum number of alarm messages to buffer. Allowed values: integers 1-1000000</td>
</tr>
<tr>
<td>-p</td>
<td>Do not drop messages</td>
<td>Drop low priority messages when buffer is full. Specifically, when -m &lt;max&gt; messages have been received but it is not yet time to write the buffer contents to the &lt;box_SMF_ac&gt; database, then low priority messages in the buffer are dropped in favour of higher priority messages that may be received on its input stream.</td>
</tr>
<tr>
<td>-r &lt;node&gt;</td>
<td>Direct to the Oracle DB</td>
<td>Rep Node - specify replication requester node.</td>
</tr>
<tr>
<td>-t &lt;secs&gt;</td>
<td>1</td>
<td>Commit Interval - maximum interval between committing to the database (seconds).</td>
</tr>
<tr>
<td>-u &lt;user/pass&gt;</td>
<td>/</td>
<td>Use the supplied Oracle user/password pair.</td>
</tr>
</tbody>
</table>

Usage examples

Here is an examples of using smsAlarmDaemon:

```
smsAlarmDaemon -l 5 -h 30 -n 10 -m 2000 -p -d -a /volB/home/saich -r 750 -u smf/smf -f -i -g -c 2 -t 2
```

- Filter Period (-l) = 5 seconds
- Heart beat (-h) = Yes every 30 seconds
- Filter Number (-n) = 10 each period
- Max number(-m) = 2000 records
- Drop low priority messages (-p) = true
- Sort messages by severity (-d) = false
- Oracle Alert Log location (-a) = /volB/home/saich
- Rep node (-r) = 750
- Oracle User (-u) = smf/smf
- Filtering (-f) = Multiple alarms combined
- Filtering type (-i) = Exact match
- GMT timezone (-g) = Yes
- Commit Rate (-c) = every 2 number of inserts
- Commit Interval (-t) = every 2 seconds if 2 records not reached

Failure

The smsAlarmDaemon on each alarm-managed node in the installation will by default generate a health-check alarm once per minute. These health check alarms will be relayed in the same fashion as all other alarms.

If these health check alarms are not received at the target destination, then the smsAlarmDaemon may have failed, and should be investigated.

Output

The smsAlarmDaemon writes error messages to the system messages file, and also writes additional output to:

```
/IN/service_packages/SMS/tmp/smsAlarmDaemonScp.log
```
smsLogCleaner

Purpose
The log cleaner looks for old/large files in the following directory:

/IN/service_packages/SMS/tmp

Any files found there, are moved into the archive subdirectory. smsLogCleaner then purges the files from the originating directory.

Startup
This task is run in the crontab for smf_oper. By default it runs at 30 minutes past each hour. It is run via the shell script:

/IN/service_packages/SMS/bin/smsLogCleanerStartup.sh

Parameters
The smsLogCleaner supports the following command-line options:

Usage:

smsLogCleaner -c <configuration file name> -d <days> -s <storage file name> [-h]

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-c &lt;configuration filename&gt;</td>
<td>logjob.conf</td>
<td>The configuration file to use.</td>
</tr>
<tr>
<td>-d &lt;days&gt;</td>
<td>7</td>
<td>How often to clean archive in days.</td>
</tr>
<tr>
<td>-s &lt;storage filename&gt;</td>
<td>storage.txt</td>
<td>The storage file to use.</td>
</tr>
<tr>
<td>-h</td>
<td></td>
<td>Provides help information.</td>
</tr>
</tbody>
</table>

At installation time, the cronjob is configured to execute by default with the following command line parameters:

- c /IN/service_packages/SMS/etc/logjob.conf
- s /IN/service_packages/SMS/tmp/sms_storage.txt
- d 7

An operator may change these values, subject to disk storage availability and site-specific archiving policies.

Failure
If the process is not running, log files in the following directory will accumulate in size and age beyond the expected values.

/IN/service_packages/SMS/tmp

Output
The smsLogCleaner run by smf_oper writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/smsLogCleaner.log

Continued on next page
The logjob.conf configuration file has the following format:

```
log <file> age <hrs> size <size> arcdir <dir> logonce
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>log &lt;file&gt;</code></td>
<td>The full directory path and name of the file to be cleaned. You can include the &quot;*&quot; wildcard in the file name if required.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>log /IN/service_packages/SMS/tmp/smsNamingServer.log</code></td>
</tr>
<tr>
<td><strong>Tip:</strong></td>
<td>Most processes and tools document where their output is written to in their Output topic.</td>
</tr>
<tr>
<td><code>age &lt;hrs&gt;</code></td>
<td>Sets the minimum age in hours for the log file before it will be cleaned. You must set either this parameter or the size parameter. If both parameters are set, then the log file is cleaned if either condition is met.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>age 100</code></td>
</tr>
<tr>
<td><code>size &lt;size&gt;</code></td>
<td>Sets the minimum size for the logfile before it will be cleaned. You must set either this parameter or the age parameter. If both parameters are set, then the log file is cleaned if either condition is met.</td>
</tr>
<tr>
<td>Examples:</td>
<td><code>size 60K</code>, or <code>size 60M</code></td>
</tr>
<tr>
<td><code>arcdir &lt;dir&gt;</code></td>
<td>The directory to use to store the old log file. If this parameter is not specified, then the log file is deleted.</td>
</tr>
<tr>
<td>Example:</td>
<td><code>arcdir /IN/service_packages/SMS/tmp/archive</code></td>
</tr>
<tr>
<td><code>logonce</code></td>
<td>Only specify this parameter if you just want to keep one archived version of the log file.</td>
</tr>
</tbody>
</table>
smsStatsDaemon

Purpose
The smsStatsDaemon program is the key component in the statistics process. The statistics process gathers and updates all statistics values through a single consistent mechanism over the network.

The smsStatsDaemon can optionally dynamically load extension libraries at runtime to provide extra functionality. This functionality includes node uptime, process uptime, and database row counts.

Startup
This task is started by entry scp3 in the inittab, via the shell script:

```
/IN/service_packages/SMS/bin/smsStatsDaemonStartup.sh
```

smsStatsDaemon configuration
The stats daemon can run in one of two modes:

- standard mode or
- legacy mode.

The standard mode uses command line options to configure the smsStatsDaemon, and uses the SMF_STATISTICS_DEFN table in the SMF database to define the statistics which should be collected.

The legacy mode is configured using a combination of command line parameters and a configuration file. The configuration file defines the statistics which should be collected.

Depending on which mode the smsStatsDaemon is running in, a different set of parameters will be used.

Parameters
The command line parameters for the smsStatsDaemon are:

Usage:

```
```

Or for legacy form:

```
```

The available parameters are:

- **-e <secs>**
  
  **Syntax:** `-e <secs>
  
  **Description:** The minimum number of seconds between logging statistic counts of zero.
  
  **Type:** Integer
  
  **Optionality:** Optional (default used if not set).
  
  **Default:** 0
  
  **Notes:** This only applies to collection mode 1 (always report).

Continued on next page
smsStatsDaemon, Continued

Parameters (continued)

- **f <dir/file>**
  - **Syntax:** -f <dir/file>
  - **Description:** The stats configuration file (full path, includes file name).
  - **Type:** String
  - **Optionality:** Optional (stats file not used if not set).
  - **Default:** null

- **-v**
  - **Syntax:** -v
  - **Description:** Use verbose mode (provide more information while processing).
  - **Type:** Boolean
  - **Optionality:** Optional (default used if not set).
  - **Default:** Do not use verbose mode.
  - **Notes:** Usually used for debugging set-up problems.

- **-d <rows>**
  - **Syntax:** -d <rows>
  - **Description:** The number of rows for dynamic stats table.
  - **Type:** Integer
  - **Optionality:** Optional (default used if not set).
  - **Default:** 500
  - **Notes:** The dynamic stats table is the shared memory hash table used to contain the statistics information. The size of this table varies with the number of statistics being collected.

- **-h <ratio>**
  - **Syntax:** -h <ratio>
  - **Description:** The ratio of the size of the hash index to the dynamic stats table.
  - **Type:** Integer
  - **Optionality:** Optional (default used if not set).
  - **Default:** 2
  - **Notes:** The dynamic stats table is the shared memory hash table used to contain the statistics information. The size of this table varies with the number of statistics being collected.

Continued on next page
smsStatsDaemon, Continued

Parameters (continued)

-F

Syntax: -F
Description: Only do USMS-specific statistic collection.
Type: Boolean
Optionality: Optional (default used if not set).
Default: Collect all statistics.
Notes: Only set when smsStatsDaemon is running on an USMS.
The SMF_STATISTICS_EXTN table specifies for each extension statistic whether it is an USMS stat or not. Also the shared memory hash table specifies whether each statistic is an USMS stat. Hence the -F option can do only USMS stats or all stats.
Examples: Uptime of smsMaster (on page 110) is an USMS stat, uptime of updateLoader (on page 150) is not an USMS stat.

-m <size>

Syntax: -m <size>
Description: The size of the details column in the dynamic stats table.
Type: Integer
Optionality: Optional (default used if not set).
Default: 80
Notes: The dynamic stats table is the shared memory hash table used to contain the statistics information.

-i

Syntax: -i
Description: Silently ignore mount points that are longer than the -m <size> (on page 143) limit.
Type: Boolean
Optionality: Optional (not used if not set).

-S

Syntax: -S
Description: Silently drop statistics where the details field is longer than -m <size> (on page 143).
Type: Boolean
Optionality: Optional (not used if not set).

Continued on next page
### Parameters for standard mode

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Description</th>
<th>Type</th>
<th>Optionality</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>-T</td>
<td>-T</td>
<td>Truncate the detail field for statistics that exceed -m &lt;size&gt; (on page 143).</td>
<td>Boolean</td>
<td>Optional (not used if not set).</td>
<td></td>
</tr>
<tr>
<td>-u &lt;usr/pwd&gt;</td>
<td>-u &lt;usr/pwd&gt;</td>
<td>The userid and password to use to log into the SMF database.</td>
<td>String</td>
<td>Optional (default used if not set).</td>
<td>/</td>
</tr>
<tr>
<td>-r &lt;node&gt;</td>
<td>-r &lt;node&gt;</td>
<td>The replication node number smsStatsDaemon should use.</td>
<td>Integer</td>
<td>Optional (default used if not set).</td>
<td>-1</td>
</tr>
<tr>
<td>-w</td>
<td>-w</td>
<td>Do Row Count statistic collection.</td>
<td>Boolean</td>
<td>Optional (not used if not set).</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** These parameters cannot be used with the Parameters for legacy mode (on page 144).

### Parameters for legacy mode

These parameters can be used if smsStatsDaemon is being used in legacy mode.

**Note:** These parameters cannot be used with the Parameters for standard mode (on page 144).
smsStatsDaemon, Continued

Parameters for legacy mode (continued)

- **c** <dir>
  
  **Syntax:** \( \text{paraMeter} = <> \)
  
  **Description:** Current statistics file directory.
  
  **Type:** String
  
  **Optionality:** Optional (default used if not set).
  
  **Default:** /tmp

- **a** <dir>
  
  **Syntax:** -d <dir>
  
  **Description:** Archived statistics file directory.
  
  **Type:** String
  
  **Optionality:** Optional (default used if not set).
  
  **Default:** /tmp

- **t** <secs>
  
  **Syntax:** -t <secs>
  
  **Description:** The maximum number of seconds a statistics file can be open.
  
  **Type:** Integer
  
  **Optionality:** Optional (default used if not set).
  
  **Default:** 1800

- **s** <Kb>
  
  **Syntax:** -s <Kb>
  
  **Description:** The maximum number of Kb a stats file can reach.
  
  **Type:** Integer
  
  **Optionality:** Optional (default used if not set).
  
  **Default:** 10

---

**Failure**

If the smsStatsDaemon fails, statistics on that UAS will not be processed. When the smsStatsDaemon is restarted the statistics will be processed.

**Output**

The smsStatsDaemon writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/smsStatsDaemon.log

**Measurement IDs - standard mode**

Measurement IDs for the statistics which should be collected are loaded from the SMF_STATISTICS_DEFN and SMF_STATISTICS_EXTN tables in the Oracle Standard DB instance given by ORACLE_SID. Setting the TWO_TASK variable allows a machine without a database instance running access a database on a remote machine. One application of this may be to allow monitoring of a remote disaster recovery machine.

*Continued on next page*
The shared memory area contains an index to the statistics measurements it contains. Each measurement has an accumulator for up to 16 SLPI instances. The single SLPI process is the only process to write to that buffer. The per-SLPI statistics counters are never reset, the smsStatsDaemon treats them as read-only.

This table gives the type and valid values of the parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENTDIR</td>
<td>256 characters</td>
<td>Where active statistics files are stored.</td>
</tr>
<tr>
<td>ARCHIVEDIR</td>
<td>256 characters</td>
<td>Where archived statistics files are stored.</td>
</tr>
<tr>
<td>OPENTIME</td>
<td>unsigned long</td>
<td>The maximum length of time a statistics file should remain open. The range of values is 1-1440 minutes.</td>
</tr>
<tr>
<td>MAXSIZE</td>
<td>unsigned long</td>
<td>The maximum size (in Kbytes) a statistics file is allowed to reach.</td>
</tr>
<tr>
<td>NOTIFY</td>
<td>256 characters</td>
<td>Space separated e-mail accounts to notify when the statistics file is rotated. This value is optional. If it does not exist, no e-mail is sent.</td>
</tr>
<tr>
<td>MID</td>
<td>Measurement description record</td>
<td>Specifies a measurement to be made available.</td>
</tr>
<tr>
<td>MERGECPUSTATS</td>
<td>1 character</td>
<td>If this config file entry is set, it overrides the default behaviour. The command line switch (-C) when set, overrides both this config file entry and the default entry.</td>
</tr>
</tbody>
</table>

To provide full backwards support for sites using the SMS version 1 style configuration, the use of a configuration file is optional. A configuration file will be searched for according to the following rules:

- If the `-f <config_file_loc>` parameter is specified, the config file is used. An error occurs if the specified file does not exist.
- If the `-f` parameter is omitted, then a search is made for a file "etc/smsStatsDaemon.cfg" or "./etc/smsStatsDaemon.cfg". If one of these files exists, then the file is used. Otherwise the smsStatsDaemon will start with the default configuration as described above.

The configuration file provides:

- Parameters (for example, max open file size, archive file directory), and
Legacy mode configuration - config file (continued)

- Measurement IDs (specified using "MID=..." entries).

**Note:** Any configuration specified in the command line will override the details in the configuration file. The database configuration is NOT used.

### Syntax for the stats_config file

For legacy sites, the syntax of the stats_config file is given.

This is an example of a stats_config file:

```plaintext
# Log file locations (trailing / is optional)
CURRENTDIR=/IN/service_packages/SMS/statistics/current
ARCHIVEDIR=/IN/service_packages/SMS/statistics/archive
# Max open time is 10 min (1-1440)
OPENTIME=10
# Max file size is 128 kb (unlimited)
MAXSIZE=128
MID=npNotFound,NP,Portability request with no target,3600,Category NP
MID=npTimeOut,NP,Exceeded regulated time-out on connect,300,No comment
MID=vpnManagement,VPN,Calls to management hotline,3600,Non-charged
MID=vpnSchedule,VPN,Calls activating scheduled routing,3600,No comment
```

**Where:**

- `#` indicates a comment. The comment character needs to start at the beginning of a line. The entire line is then ignored (up to 255 characters).
- `CURRENTDIR` indicates the working directory of the daemon. This is where a temporary statistics file is stored until the file size of open time exceeds the configured value.
- `ARCHIVEDIR` is the location a closed statistics file is moved to.
- `OPENTIME` indicates how long a statistics file can stay active (that is, how long can the daemon keep on writing statistics into a file) in minutes.
- `MAXSIZE` is the maximum allowable size of a statistics file in Kilobytes.

**Note:** If a statistics file becomes overloaded half way through a dump, the entire record will still be written.

- `MID` indicates the measurements to retrieve. Refer to the subsection on Measurements.

*Continued on next page*
### Measurements

Measurements are specified in the configuration file using one MID command for each measurement to be defined.

MID commands are comma separated value names, that must be in the order below:

- ID
- APPLICATION
- DESCRIPTION
- PERIOD
- COMMENT
- EXTN
- KEYWORD
- DETAIL
- LIB_NAME
- FUNCTION_NAME

### Examples:

An example MID line without the extension fields might be:

```
MID=vpnManagement,VPN,Calls to management hotline,3600,Non-charged
```

An example MID line with the extension fields would be:

```
MID=STATSDAEMON,SCP_SYSTEM,Uptime for smsStatsDaemon process,60,smsStatsDaemon process uptime in minutes,EXTN,PROCESS_UPTIME,smsStatsDaemon,libsmsextrastats.so,getProcessUptime
```

This table describes the measurement parameters.

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>20 characters</td>
<td>The application ID. This may be up to 20 characters for clarity, however the first three characters must be unique.</td>
</tr>
<tr>
<td>COMMENT</td>
<td>256 characters</td>
<td>Textual comment relating to the statistic.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>256 characters</td>
<td>The textual description of the measurement.</td>
</tr>
<tr>
<td>DETAIL</td>
<td>80 characters</td>
<td>Extra data required to measure stat, i.e. process name for process uptime stats. Can be NULL (empty string)</td>
</tr>
<tr>
<td>EXTN</td>
<td>5 characters</td>
<td>The keyword 'EXTN' indicating that this mid line has the extra fields</td>
</tr>
<tr>
<td>FUNCTION_NAME</td>
<td>50 characters</td>
<td>Function within the library (specified at LIB_NAME) to call in order to measure stat.e.g. getNodeUptime.</td>
</tr>
<tr>
<td>ID</td>
<td>20 characters</td>
<td>The measurement ID.</td>
</tr>
</tbody>
</table>
### Legacy mode configuration - config file (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>KEYWORD</td>
<td>20 characters</td>
<td>May be required by measurement function. e.g. UPTIME_NODE.</td>
</tr>
<tr>
<td>LIB_NAME</td>
<td>30 characters</td>
<td>Dynamic library to load to get stat measurement function. e.g. libsmsextrastats.so.</td>
</tr>
<tr>
<td>PERIOD</td>
<td>unsigned long</td>
<td>The time in seconds between each recording in the output file of this statistic. Value range is 10-31536000 (1yr).</td>
</tr>
</tbody>
</table>

After a change is made to the Measurement IDs, the smsStatsDaemonRep process needs to be notified via a SIGHUP. This can be performed manually, or via the smsStatsDaemonRepReload.sh script provided as part of the installation.

### Updating smsStatsDaemon measurements

After a change is made to the Measurement IDs, either via the database, or via modifying "MID=..." entries in the stats_config file, the smsStatsDaemon process needs to be notified via a SIGHUP. This can be performed manually, or via the smsStatsDaemonReload.sh script provided as part of the installation.
updateLoader

**Purpose**

The updateLoader accepts updates from the smsMaster and makes the requested changes to the database it is configured to update. More than one updateLoader may run on each UAS. For more information about updateLoaders and replication, see *What is Replication?* (on page 22)

If the UAS data becomes out of sync with the data in the SMF, a resync can be done to ensure the UAS has the correct information. It should not be necessary to do a manual resync. The system does automatic resyncs as necessary.

There are three cases where the system will resync:

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Node is Out of History</td>
<td>Where a node is isolated SMS will hold a queue of updates for the node. In the replication.def file there is a max pending variable that gives the maximum number of updates that will be held in this queue for each UAS. If this limit is exceeded (the node is out of history) USMS will drop all the entries and force a resync of the node when it comes back on line.</td>
</tr>
<tr>
<td>New Node Added</td>
<td>When a new node is added to the system a replication config file will be sent to the node. This forces a resync.</td>
</tr>
<tr>
<td>Replication Changed</td>
<td>If the replication config file for a node is changed then a resync will be forced.</td>
</tr>
</tbody>
</table>

**Startup**

This task is started by entry scp5 in the initab, via the shell script:

```
/IN/service_packages/SMS/bin/updateLoaderStartup.sh
```

**Parameters**

The updateLoader supports the following command-line options:

**Usage:**

```
updateLoader [-nodeid <node number>] [-resync]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-nodeid &lt;node number&gt;</td>
<td>274</td>
<td>The node number of the updateLoader requesting the resync.</td>
</tr>
<tr>
<td>-resync</td>
<td></td>
<td>Causes the updateLoader to re-synchronise with the smsMaster.</td>
</tr>
</tbody>
</table>

**Failure**

If the updateLoader is not working, updates from the USMS to the UAS database will be unsuccessful. The UAS will continue to run on the last configuration successfully loaded from the USMS.

An error message will be logged to the syslog and the updateLoader log, and may be logged to the smsMaster log.

**Output**

The updateLoader writes error messages to the system messages file.
Chapter 8

Tools and Utilities

Overview

Introduction
This chapter provides a description of the operational programs or executables used by the system.

Executables are located in the /IN/service_packages/SMS/bin directory.

Some executables have accompanying scripts that run the executables after performing certain cleanup functions. All scripts should be located in the same directory as the executable.

In this chapter
This chapter contains the following topics.

- cmnConfigSyntaxCheck .......................................................... 152
- compareNode .......................................................................... 153
- infoDisplayer .......................................................................... 154
- inputBootstrap ........................................................................ 155
- repConfigWrite ....................................................................... 157
- resyncServer ......................................................................... 158
- smsCompareResyncClient .................................................... 159
- smsCompareResyncServer ..................................................... 162
- smsDumpRepConfig ................................................................ 175
- smsLogTest ............................................................................ 177
- smsManualRequester ............................................................ 178
- smsProcessCdr ....................................................................... 179
- smsRecordStatistic ............................................................. 186
- startMerge ............................................................................. 187
cmnConfigSyntaxCheck

**Purpose**
cmnConfigSyntaxCheck is used to check that the syntax of the eserv.config file is correct.

**Configuration**

cmnConfigSyntaxCheck accepts the following command line options.

**Usage:**

```
cmnConfigSyntaxCheck [-v -d] filename [filename [...]]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-v</td>
<td>-</td>
<td>Verbose mode. Displays in detail all information that is available.</td>
</tr>
<tr>
<td>-d</td>
<td>-</td>
<td>Reads and dumps the named files</td>
</tr>
</tbody>
</table>

**Output**

cmnConfigSyntaxCheck displays the results of the syntax check on the terminal.

**Example:** This text shows an example of a report from the cmnConfigSyntaxCheck.

```
$ cmnConfigSyntaxCheck -v filename /ACS/etc/acs.conf

cmn::FileNotFoundException: Opening config file filename

Config file syntax error: ACS/etc/acs.conf:30: Syntax Error
```
compareNode

Purpose
This command can be used to initiate a full database comparison of an UAS database with the definitive copy in the SMF database.

This is used to ensure that an UAS database has all its data consistent with the SMF database. Under normal conditions, this should always be the case, but there may be a time (for example, after multiple failures) where the System Administrator wants to check that an UAS database is consistent.

The compareNode tool requests a comparison between the contents of the SMF database and one other node, by invoking comparisonServer. This is a more time-efficient method than a resync. All the entries of all the tables that are defined to be replicated to the specified updateLoader will be compared.

A full report of the comparison is written in the report directory (REPORT DIR) on the UAS machine.

Configuration
compareNode accepts the following command line options.

Usage:
compareNode [ -hostname <hostname> | -master <node num> ] [ -with <node num> ] [ -timeout <seconds> ]

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-hostname</td>
<td></td>
<td>Sets the hostname of the superior node in the comparison. (Optional. If used, -master must = 0, that is, if -master must be set to off.)</td>
</tr>
<tr>
<td>-master</td>
<td>1</td>
<td>Node number of the superior node in the comparison. (Optional, as the default will be used if it is not set. Or it can be turned off by setting to 0, and a hostname specified instead.)</td>
</tr>
<tr>
<td>-with</td>
<td>256</td>
<td>Node number of the updateLoader in the comparison. (Optional, as the default will be used if it is not set.)</td>
</tr>
<tr>
<td>-timeout</td>
<td>10</td>
<td>Number of seconds before the connection between the nodes in the comparison is timed out. (Optional, as the default will be used if it is not set.)</td>
</tr>
</tbody>
</table>

Note: This can be an infMaster.

Example: This text shows the common usage of compareNode being run on the superior master in a node comparison.

```
compareNode -with 301
```

Failure
If compareNode fails, it will send error messages to stdout and syslog.

Output
The compareNode writes error messages to the system messages file, and also writes additional output to:

```
/IN/html/SMS/output/<node number>/<timestamp>.html
```
infoDisplay

Purpose
infoDisplay is an executable which can be used to display Update Request Information results.

Configuration
infoDisplay supports the following command-line options:

Usage:
infoDisplay [-host value] [-master value] [-nodeid value] [-timeout value]

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>host</td>
<td>localhost</td>
<td>The local host name</td>
</tr>
<tr>
<td>master</td>
<td>0</td>
<td>The node number of the smsMaster</td>
</tr>
<tr>
<td>nodeID</td>
<td>1</td>
<td>The ID of the node for which the information is to be displayed.</td>
</tr>
<tr>
<td>timeout</td>
<td>10</td>
<td>Period after which infoDisplay will timeout</td>
</tr>
</tbody>
</table>

Output
Examples:
bash-2.05$ .infoDisplay -nodeid 999 -master 1
 initialiseNode: Reading
 '/IN/service_packages/SMS/etc/replication.def'
 initialiseNode: heartbeatPeriod 20
 initialiseNode: heartbeatTimeout 20
 initialiseNode: connectionTimeout 2
 initialiseNode: masterPortNum 12343
 initialiseNode: queueWarnThresh 5
 initialiseNode: queueErrThresh 100000
 initialiseNode: queueCritThresh 1000000
 initialiseNode: hBTolerance 10.0
 initialiseNode: commitIdleTime 0.100000
 initialiseNode: commitBusyTime 10.0
 initialiseNode: tcpAbortSecs 20
 initialiseNode: oracleUserPass '/'
 initialiseNode: reportDir '/IN/service_packages/SMS/tmp/'
 initialiseNode: statusFile '/IN/html/status.html'
 initialiseNode: configFilePath
 '/IN/service_packages/SMS/etc/replication.config'
 initialiseNode: configFileName 'replication.config'
 initialiseNode: node number 999
 initialiseNode: node type 5
 initialiseNode: s side updates 1
Nov 22 22:05:17 infoDisplay(6589) NOTICE: Master Controller
'./infoDisplay' process started (node 999)
**inputBootstrap**

### Purpose

The purpose of the inputbootstrap binary is to produce a configuration file to be passed to the smsCompareResyncServer from replication.config. It is started by the comparisonServer or the resyncServer which initiates requests. It can also be executed manually from the command line.

It must be noted that this binary cannot be run with DEBUG when used with the comparisonServer. (Applicable to production environment).

**Note:** This binary is not intended to be run by the User. Please contact your Oracle support before attempting to do so.

### Configuration

inputBootstrap accepts the following command line options.

**Usage:**

```
```

Or long form:

```
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n, --node-id</td>
<td>none</td>
<td>The replication node ID for which the configuration file is produced. (Required.) Allowed values: integers, (any signed number of reasonable value, usually in decimal/decimal).</td>
</tr>
<tr>
<td>-c, --config-file</td>
<td>/IN/service_packages/SMSS/etc/replication.config</td>
<td>Name of the replication configuration file to use. (Optional.) Allowed values: string</td>
</tr>
<tr>
<td>-a, --address</td>
<td>from file specified in config-file</td>
<td>The IP address of the node. This cannot be specified if the ‘--hex-address’ option is specified. (Optional.) If specified, this will override all addresses specified in the configuration file. Allowed values: string</td>
</tr>
<tr>
<td>--hex-address</td>
<td>from file specified in config-file</td>
<td>The IP address of the node as a hex string. This cannot be specified if the ‘-a’ (‘--address’) option is specified. (Optional.) If specified, this will override all addresses specified in the configuration file. Allowed values: string</td>
</tr>
</tbody>
</table>

*Continued on next page*
### Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-p, --port</td>
<td>none</td>
<td>The port number to connect to at the given node. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This can only be used when the address is also specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong> integers, (any signed number of reasonable value, usually in decimal/octal/hex).</td>
</tr>
<tr>
<td>-r, --preserve-ranges</td>
<td>false when missing</td>
<td>Leave the data from the configuration file as it is and do not correct the group ranges. This option is implied by the -e option.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>-u, --oracle-user</td>
<td>smf/smf</td>
<td>The Oracle database connection string (userid/password). (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• string</td>
</tr>
<tr>
<td>-e, --enhanced-recovery</td>
<td>false when missing</td>
<td>Restrict range of rows to resync. If enhanced recovery mode is possible, the smsMaster process sets this option.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>-h, --help</td>
<td>false when missing</td>
<td>Shows the help for this binary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>-b, --build-info</td>
<td>false when missing</td>
<td>Prints out program build information of the binary.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
</tbody>
</table>

**Note:** Long options can be separated from their values by an equal sign (=), or you can pass the value as the following argument on the command line (e.g., '--port 4000' or '--port=4000'). Short options must have their values passed after them on the command line, and in the case of boolean short options, cannot have values (they default to true) (e.g., '-p 4000' or '-f').

### Failure

If `inputBootstrap` fails, it will send error messages to stdout and syslog.
repConfigWrite

Purpose

This command can be used to create a replication.config file. It reads the local database to obtain the replication set-up and writes the file to the directory specified by the output parameter.

Generally this function is performed via the SMS Java screens.

For more information, see:

- replication.config File (on page 39)
- smsDumpRepConfig (on page 175), and
- SMS User's Guide.

Startup

This task is started by clicking Create Config File on the Table Replication tab of the Node Management screen.

It can also be started on the USMS from the command line.

For more information about the Node Management screen, see SMS User's Guide.

Configuration

repConfigWrite accepts the following command-line options:

Usage:

repConfigWrite [-user <user/password>] [-output <file>]

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-user</td>
<td>Oracle user/password for the SMF. Example: smf/smf</td>
<td>Oracle user/password for the SMF. Example: smf/smf</td>
</tr>
<tr>
<td>-output</td>
<td>./repConfigNNNNN</td>
<td>The output path and filename for the replication.config file. (Optional.) Example: /IN/service_packages/SMS/etc/replication.config</td>
</tr>
<tr>
<td></td>
<td>(Where NNNNN is a version number that counts for the number of times the file has generated OK.)</td>
<td>(Where NNNNN is a version number that counts for the number of times the file has generated OK.)</td>
</tr>
</tbody>
</table>

Failure

If repConfigWrite fails, replication.config may not have been written correctly. You can check the content of replication.config with smsDumpRepConfig. If there is a problem with replication.config, replication will not work.

Output

The repConfigWrite writes error messages to the system messages file, and also writes additional output to the specified directory and file.
resyncServer

Purpose
resyncServer initiates resyncs between databases by sending a resync request to a node Master process. This overwrites data in an UAS with data from the SMF. The node Master process is usually smsMaster.

This process is started by the smsMaster when a database resync is required and runs only for the duration of the resync. It should not be run manually.

Configuration
resyncServer accepts the following command line options.

Usage:
resyncServer <inf node> <address> <port> <enhanced_recovery>

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;inf node&gt;</td>
<td></td>
<td>The node with the database which will be updated.</td>
</tr>
<tr>
<td>&lt;address&gt;</td>
<td></td>
<td>The ip address or hostname of the node which will be updated.</td>
</tr>
<tr>
<td>&lt;port&gt;</td>
<td></td>
<td>The port number on the node which will be updated.</td>
</tr>
<tr>
<td>&lt;enhanced_recovery&gt;</td>
<td>off</td>
<td>If set to on, the number of rows in the inferior database will not be counted during the resync. Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• on, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• off.</td>
</tr>
</tbody>
</table>

Output
resyncServer writes error messages to the system messages file, and also writes additional output to:

/IN/service_packages/SMS/tmp/resyncServer.log
**smsCompareResyncClient**

**Purpose**
This is a child process of updateLoader. It is called by smsCompareResyncServer and updates the UAS during replication on a clustered install. It also performs database resynchronisations and comparisons on the inferior node during replication checks.

This process is not intended to be be started manually. It is installed on the UAS.

**Configuration**

smsCompareResyncClient accepts the following command line options.

**Usage:**

```
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n, --node-id</td>
<td>The node number of the client. (Required.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Allowed values: integer (any signed number of reasonable value, usually in decimal/octet/hex)</td>
<td></td>
</tr>
<tr>
<td>-u, --oracle-user</td>
<td>smf/smf</td>
<td>The Oracle database connection string. (Optional.)</td>
</tr>
<tr>
<td></td>
<td>Allowed values: string</td>
<td></td>
</tr>
<tr>
<td>-h, --help</td>
<td>false</td>
<td>Prints this help screen. (Optional.)</td>
</tr>
<tr>
<td></td>
<td>Allowed values:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1, on, yes, true, or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 0, off, no, false.</td>
<td></td>
</tr>
<tr>
<td>-b, --build-info</td>
<td>false</td>
<td>Prints out program build information then exits. (Optional.)</td>
</tr>
<tr>
<td></td>
<td>Allowed values:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1, on, yes, true, or</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 0, off, no, false.</td>
<td></td>
</tr>
<tr>
<td>-i, --inform-parent</td>
<td>no process is informed</td>
<td>Process to inform when the process is in a position to resync/compare. For use only when used from the updateLoader process. (Optional.)</td>
</tr>
<tr>
<td></td>
<td>Allowed values: integers (any signed number of reasonable value, usually in decimal/octet/hex)</td>
<td></td>
</tr>
<tr>
<td>-p, --port</td>
<td>The port to listen on for connections from the server. (Optional.)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Allowed values: integers (any signed number of reasonable value, usually in decimal/octet/hex)</td>
<td></td>
</tr>
</tbody>
</table>

*Continued on next page*
### Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-o, --tcp-timeout</td>
<td>-1</td>
<td>Timeout (in seconds) on TCP connection. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• positive integers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• -1 = no timeout</td>
</tr>
<tr>
<td>-t, --throttle-cpu</td>
<td>false</td>
<td>To throttle the client. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>--outside-throttle-sample-rate</td>
<td></td>
<td>Sample rate in seconds for throttling while not throttling. (Required if -t given.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• integers (any signed number of reasonable value, usually in decimal/octet/hex)</td>
</tr>
<tr>
<td>--inside-throttle-sample-rate</td>
<td></td>
<td>Sample rate in seconds for throttling while throttling. (Required if -t given.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• integers (any signed number of reasonable value, usually in decimal/octet/hex)</td>
</tr>
<tr>
<td>--start-threshold</td>
<td></td>
<td>CPU usage percentage to start throttling at. (Required if -t given.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• integers (any signed number of reasonable value, usually in decimal/octet/hex)</td>
</tr>
<tr>
<td>--stop-threshold</td>
<td></td>
<td>CPU usage percentage to end throttling at. (Required if -t given.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• integers (any signed number of reasonable value, usually in decimal/octet/hex)</td>
</tr>
<tr>
<td>-s, --storage-dir</td>
<td>/tmp</td>
<td>Directory to store required database changes in during a resync. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• string</td>
</tr>
<tr>
<td>--database-write-buffer-size</td>
<td>10</td>
<td>The size in terms of records of the buffer size for writing to the database. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• integers (any signed number of reasonable value, usually in decimal/octet/hex)</td>
</tr>
<tr>
<td>--database-read-buffer-size</td>
<td>100</td>
<td>The size in terms of records of the buffer size for reading to the database. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• integers (any signed number of reasonable value, usually in decimal/octet/hex)</td>
</tr>
</tbody>
</table>

*Continued on next page*
## Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--database-commit-period</td>
<td>1000</td>
<td>The number of changes to make to the database before committing them to the database. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong> integers (any signed number of reasonable value, usually in decimal/octal/hex)</td>
</tr>
<tr>
<td>--max-buffer-size</td>
<td>approximately 50Mb</td>
<td>The maximum size (in bytes) to allow messages received from the server to be. This size is reflected in the maximum size of bytes to allocate in memory for such messages. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong> integers (any signed number of reasonable value, usually in decimal/octal/hex)</td>
</tr>
<tr>
<td>--dump-core-instead-of-exception</td>
<td>false</td>
<td>If set, will force the process to dump a core file (and exit) if any network messages are received bigger than max-buffer-size.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong> 1, on, yes, true, or 0, off, no, false.</td>
</tr>
<tr>
<td>--long-raw-size</td>
<td>512K</td>
<td>The maximum size (in bytes) to allocate for a long raw field used in a resync.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong> 512K.</td>
</tr>
</tbody>
</table>

**Note:** Long options can be separated from their values by an equal sign (=), or you can pass the value as the following argument on the command line (for example, `--node-id 257` or `--node-id=257`). Short options must have their values passed after them on the command line, and in the case of boolean short options, cannot have values (they default to true) (for example, `-p 4000` or `-t`).
smsCompareResyncServer

Purpose

smsCompareResyncServer performs comparisons and resyncs of data in specified tables and replication groups. This enables you to:

- check that replication is working correctly, or
- force updates of data between nodes.

Note: This process is usually started by resyncServer. It is not intended to be run manually.

This is installed on the USMS.

Configuration

smsCompareResyncServer accepts the following command line options.

Usage:

```
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--dump-core-instead-of-exception</td>
<td>false when missing</td>
<td>If set, will force the process to dump a core file (and exit) if any network messages are received bigger than max-buffer-size.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>--max-buffer-size</td>
<td>approximately 50Mb</td>
<td>The maximum size (in bytes) to allow messages sent to the client to be. This size is reflected in the maximum size of bytes to allocate in memory for such messages.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: integers (any signed number of reasonable value, usually in decimal/octal/hex).</td>
</tr>
<tr>
<td>-d, --dont-count-rows</td>
<td>false when missing</td>
<td>Do not make a count of the rows in the database. For very large comparisons/resyncs this may give a speed improvement.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>-m, --inform-master</td>
<td>false when missing</td>
<td>If performing a resync, inform the smsMaster.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
</tbody>
</table>

Continued on next page
smsCompareResyncServer, Continued

### Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>--database-read-buffer-size</td>
<td>100</td>
<td>The size (in records) of the buffer size for reading from the database. Must match the --database-write-buffer-size specified for the smsCompResyncClient. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: integers (any signed number of reasonable value, usually in decimal/octet/hex).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> Performance of the compare/resync can be seriously impacted if the number specified is too low, a recommended value for this parameter is 1000+.</td>
</tr>
<tr>
<td>--cancel-on-eof</td>
<td>false when missing</td>
<td>Tells the server to cancel the resync/compare when EOF on standard input is reached.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> This option is specifically for the inetCompareServer setup.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>--use-ip</td>
<td>none</td>
<td>Forces the server to use the IP address ranked as per the mentioned integer for each client. If there are not enough IP addresses listed, or this option is not specified, it will start from the first IP address, attempting each in turn until connected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: integers (any signed number of reasonable value, usually in decimal/octet/hex).</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Examples:</strong> If --use-ip = 2, the server will use the second IP address listed.</td>
</tr>
<tr>
<td>-r, --report-directory</td>
<td>/IN/html/output/SMS/</td>
<td>The base directory to create and store final reports in. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: string.</td>
</tr>
<tr>
<td>-o, --tcp-timeout</td>
<td>0</td>
<td>Timeout (in seconds) on TCP connection. (Optional.) Zero = no timeout.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: integers (any signed number of reasonable value, usually in decimal/octet/hex).</td>
</tr>
<tr>
<td>-i, --input-file</td>
<td>Input is expected on the standard input stream.</td>
<td>Contains the name of a configuration file as input information for performing a resync/compare. (Optional.) See Input file for details.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: string.</td>
</tr>
</tbody>
</table>

Continued on next page
smsCompareResyncServer, Continued

### Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-u, --oracle-user</td>
<td>smf/smf</td>
<td>The Oracle database connection string. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong> string.</td>
</tr>
<tr>
<td>-b, --build-info</td>
<td>false when missing</td>
<td>Prints out program build information then exits.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>-h, --help</td>
<td>false when missing</td>
<td>Prints this help screen.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true, or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false.</td>
</tr>
<tr>
<td>--long-raw-size</td>
<td>512K</td>
<td>The maximum size (in bytes) to allocate for a long raw field used in a resync.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 512K.</td>
</tr>
</tbody>
</table>

**Notes:**
- All options apart from `-h`, `-b` and `-i` can be specified in the input configuration.
- Long options can be separated from their values by an equal sign (`=`), or you can pass the value as the following argument on the command line (for example, `--tcp-timeout 100` or `--tcp-timeout=100`). Short options must have their values passed after them on the command line, and in the case of boolean short options, cannot have values (they default to true) (for example, `-o 100` or `-h`).

### Input file
These are the configuration parameters contained within the input file optionally used by smsCompareResyncServer.

**Replication**

**Syntax:**

```
Replication = {<list of replication parameters>}
```

**Description:**

The Replication section lists the required replication parameters.

**Type:** List

**Optionality:** Mandatory

**Default:** none

Continued on next page
smsCompareResyncServer, Continued

Input file (continued)

perform

Syntax: perform = "<action>"
Description: The action to be taken by smsCompareResyncServer.
Type: Boolean
Optionality: Mandatory
Allowed:
• resync
• compare
Default: none
Example: perform = "resync"

report-row-number-limit

Syntax: report-row-number-limit = <max value>
Description: For a comparison, the maximum number of differences in a group that will be reported.
For a resync, the maximum number of errors to report of each group synchronised.
Type: Integer
Optionality: Optional (default used if not set).
Default: -1
Notes: Specifying -1 indicates no limit will be imposed.
Example: report-row-number-limit = 100

produce-final-reports

Syntax: produce-final-reports = <true|false>
Description: Whether or not to produce final reports.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true
false
Default: true (reports produced)
Example: produce-final-reports = false

Continued on next page
smsCompareResyncServer, Continued

Input file (continued)

report-directory
Syntax: report-directory = "<directory>"
Description: Specify which directory reports are to be written to.
Type: String
Optionality: Optional (default used if not set).
Allowed: Any valid directory
Default: /IN/html/output/SMS
Example: report-directory = ".">

report-after
Syntax: report-after = {
    type = "[comparisons|seconds]"
    count = <number>
}
Description: Depending on the type option:
Specify how many comparisons the client should process
before sending progress report to the server.
Specify how many seconds the client should allow to elapse
before sending a progress report to the server.
Type: List
Optionality: Optional (default used if not set).
Default: No progress reports are provided.
Notes: type and count are both mandatory when report-after is
specified.
Example: report-after = {
    count = 10
    type = "seconds"
}

stop-on-limit
Syntax: stop-on-limit = <true|false>
Description: A flag to tell the server to stop a resync or comparison for a
replication group once the report-row-number-limit (on page
165) is reached.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true
false
Default: false (don't stop)
Example: stop-on-limit = false

Continued on next page
Input file (continued)

view

Syntax:  view = {}

Description:  Describes the nodes, tables and groups for the replication action.

Type:  List

Optionality:  Mandatory

Default:  none

Node

Syntax:  Node {

  id = <number>

  address = [<string>, <string>, ...]

}

Description:  A list of nodes for the replication action to work on.

Type:  Array

Optionality:  Mandatory

Default:  none

id

Syntax:  id = <id>

Description:  The ID of the node to be used.

Type:  Integer

Optionality:  Optional (default used if not set).

Default:  Values from replication node configuration

Notes:  For normal replication resynchronization, these are calculated using the replication node configuration.

When running from the command line this must match the --node-id of a smsCompareResyncClient.

Example:  id = 301

Continued on next page
smsCompareResyncServer, Continued

Input file (continued)

address

Syntax: address = [<string>, <string>, ...]

Description: An array of IP addresses and port number(s) for this node ID.

Type: Array

Optionality: Mandatory

Default: none

Example: address = ["127.0.0.1:4000"
]

tables

Syntax: tables = [

  {
    table = <string> [groups-cover-table-on-scp = <bool>
    ]
    key-columns = [<str>, <str>, ...]
    other-columns = [<str>, <str>, ...]
  }
]

Description: An array of tables to action for this node ID.

Type: Array

Optionality: Mandatory

Default: none

table

Syntax: table = "<str>"

Description: The name of the table to be used in this operation.

Type: String

Optionality: Mandatory

Default: none

Example: table = "TEST_REP"

Continued on next page
smsCompareResyncServer, Continued

Input file (continued)

groups-cover-table-on-scp
Syntax: groups-cover-table-on-scp = <true|false>
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: • true
• false
Default: false
Example: groups-cover-table-on-scp = true

key-columns
Syntax: key-columns = [<str>, <str>, ...]
Description: An array of the table column keys which are to be used for this table and this operation.
Type: Array
Optionality: Optional (default used if not set).
Allowed: Any valid key column name for this table.
Default: Pre-configured values.
Notes: Normal replication resyncs are pre-configured and restricted to a maximum of 3 keys. When running from the command line this restriction is lifted. These columns must exist on the remote platform.
Example: key-columns = ["NUMBER_3"]

Continued on next page
Input file (continued)

other-columns
Syntax: `other-columns = [ ...

<str>, <str>, ...
]

Description: An array of the non keyed columns which are to be used for this table and this operation.
Type: Array
Optionality: Optional (default used if not set).
Allowed: Any valid non keyed column name for this table.
Default: Pre-configured values.
Notes: For normally replication resyncs these are pre-configured. These columns must exist on the remote platform.
Example: `other-columns = [ "VARCHAR_1","LONG_RAW_2","CHAR_4", "DATE_5"]`

Groups
Syntax: `groups = [ ...

{ ...

group = <str>

table = <str>

ranges = [ ...

<rangeData>, <rangeData>, ...
]

nodes = [ ...

<int>, <int>, ...
]

}

]

Description: An array of groups associated with this operation.
Type: Array
Optionality: Mandatory
Default: none

Continued on next page
smsCompareResyncServer, Continued

Input file (continued)

**group**

Syntax:  group = "<str>"

Description:  The name associated with a node.

Type:  String

Optionality:  Mandatory

Allowed:  Any character

Default:  none

Example:  group = "TEST_REP_0"

**table**

Syntax:  table = "<str>"

Description:  The name of the table associated with this group.

Type:  String

Optionality:  Mandatory

Allowed:  Any characters

Default:  none

Example:  table = "TEST_REP"

Continued on next page
Input file (continued)

ranges
Syntax:  ranges = [
        {
          from = [<from data>]
          to = [<to data>]
        }
      ]
Description: An array specifying the ranges to be associated with this group, table, and for this node.
Type: Array
Optionality: Mandatory
Default: none
Notes: Although an index to support the ranges specified is not required, it is recommended an index is used for performance reasons.
The number of elements in the from and to conditions must be the same and must match tables, key-columns entry for the table specified.
Example: ranges = [
        {
          from = [
            "1"
          ]
          to = [
            "2"
          ]
        }
      ]

nodes
Syntax:  nodes = [ <int>, <int>, ...]
Description: The list of nodes to be associated with this group.
Type: Array
Optionality: Mandatory
Allowed: Any nodes defined in Node section.
Default: none
Notes: These nodes must have been defined already in the Node section.
Example: nodes = [ 301 ]
smsCompareResyncServer, Continued

Input file example

This is an example of what the input configuration will look like, the indentation format is for readability.

```
replication = {
    perform = "resync"
    report-row-number-limit = 100
    produce-final-reports = true
    report-directory = "",
    report-after = {
        count = 10
        type = "seconds"
    }
    stop-on-limit = false
}
view = {
    nodes = [
        {id = 400
        address = [
            "127.0.0.1"
        ]
    }
    tables = [
        {table = "TEST_REP"
        groups-cover-table-on-scp = true
        key-columns = [
            "NUMBER_3"
        ]
        other-columns = [
            "VARCHAR_1",
            "LONG_RAW_2",
            "CHAR_4",
            "DATE_5"
        ]
    }
    groups = [
        {group = "TEST_REP_0"
        table = "TEST_REP"
        ranges = [
            {from = [
                "1"
            ]
            to = [
                "2"
            ]
        ]
        nodes = [
            400
        ]
    }
}
```

Output

smsCompareResyncServer writes error messages to the system messages file.

Continued on next page
**smsCompareResyncServer, Continued**

**Output (continued)**

smsCompareResyncServer writes replication checks and database comparisons to the following directory:

/IN/html/output/SMS/compare/<inferior node number>/
smsDumpRepConfig

Purpose

smsDumpRepConfig parses and displays the contents of replication.config. This provides access to the contents of the binary file where the replication configuration data is held.

For more information, see replication.config File (on page 39).

Configuration

The smsDumpRepConfig supports the following command-line options:

Usage:

smsDumpRepConfig -f <file name> [-v]

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-f &lt;file name&gt;</td>
<td>/IN/service_packages/SMS/etc/replication.config</td>
<td>Location of the configuration file to be displayed.</td>
</tr>
<tr>
<td>-v</td>
<td></td>
<td>Verbose, displays extra information, including column and field names.</td>
</tr>
</tbody>
</table>

Failure

If smsDumpRepConfig fails, it will send error messages to stdout and syslog. If an error is displayed while parsing a replication.config file, the file may be corrupted.

Output

smsDumpRepConfig displays output to stdout.

Example:

This text is an example of the output from a simple replication.config file which includes SMS and ACS replication groups between nodes 1 and 301.

smsDumpRepConfig: File
/IN/service_packages/SMS/etc/replication.config
smsDumpRepConfig: (PAD = 0)
smsDumpRepConfig: Short listing. Use -v (verbose) for full listing
-------------------------------------------------------------------
-----------
smsDumpRepConfig: Table, Column, Group definitions...
-------------------------------------------------------------------
-----------
TABLE [ACS_CALL_PLAN]
TABLE [ACS_CALL_PLAN_PROFILE]
TABLE [ACS_CALL_PLAN_STRUCTURE]
TABLE [ACS_CLI_CALL_PLAN_ACTIVATION]
TABLE [ACS_CUSTOMER]
TABLE [ACS_CUSTOMER_CLI]
TABLE [ACS_CUSTOMER_SN]
TABLE [ACS_FN_TYPE]
TABLE [ACS_GLOBAL_PROFILE]
TABLE [ACS_LANGUAGE]
TABLE [ACS_NETWORK_KEY]
TABLE [ACS_SN_CALL_PLAN_ACTIVATION]
TABLE [SMF_ALARM_MESSAGE]
TABLE [SMF_STATISTICS]
TABLE [SMF_STATISTICS_DEFN]
-------------------------------------------------------------------
-----------
smsDumpRepConfig: Replication Groups configured for each node...
-------------------------------------------------------------------
-----------
NODE NUMBER [1] Prim (192.168.0.144) Sec (0.0.0.0)
**smsDumpRepConfig**, Continued

---

**Output (continued)**

<table>
<thead>
<tr>
<th>NODE NUMBER [301] Prim (192.168.0.142) Sec (0.0.0.0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GROUP [ACS_CUSTOMER] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_FN_TYPE] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_CALL_PLAN_PROFILE] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_CALL_PLAN] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_CUSTOMER_CLI] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_CUSTOMER_SN] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [SMF_STATISTICS_DEFN] [Prim=-1] Min=('!','!',') Max=('<del>','</del>','')</td>
</tr>
<tr>
<td>GROUP [ACS_CLI_CALL_PLAN_ACTIVATION] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_GLOBAL_PROFILE] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_LANGUAGE] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_NETWORK_KEY] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
<tr>
<td>GROUP [ACS_SN_CALL_PLAN_ACTIVATION] [Prim=-1] Min=('+0',',' ') Max=('+9',',' ')</td>
</tr>
</tbody>
</table>

**Note:** Both nodes are primaries for their groups and have no secondary network address configured.
**smsLogTest**

**Purpose**

smsLogTest generates an alarm and writes it to the syslog on the local machine. You can configure the alarm details.

**Configuration**

smsLogTest supports the following command-line options:

**Usage:**

```
./smsLogTest <name> <severity> <message> [copies] [alarm_type_id]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;name&gt;</td>
<td>none</td>
<td>The subsystem name/identifier. (Required.)</td>
</tr>
<tr>
<td>&lt;severity&gt;</td>
<td>none</td>
<td>The severity value. (Required.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N or 0 [Notice]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>W or 1 [Warning]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>E or 2 [Error]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>C or 3 [Critical]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- or 4 [clear]</td>
</tr>
<tr>
<td>&lt;message&gt;</td>
<td>none</td>
<td>The message to log. (Required.)</td>
</tr>
<tr>
<td>[copies]</td>
<td>1</td>
<td>Number of times to generate the alarm. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: integer</td>
</tr>
<tr>
<td>[{alarm_type_id}]</td>
<td>none</td>
<td>Optional Alarm Type ID associated with the message. This must be up to a 9-digit number between braces (e.g. {123456789})</td>
</tr>
</tbody>
</table>

**Examples:**

```
./smsLogTest smsAlarmRelay %d "Failed to connect to Oracle" 4
{123456789}
./smsLogTest smsMaster %d "Startup Successful"
```

**Failure**

If smsLogTest fails, no alarm will be generated.

**Output**

smsLogTest displays progress and errors to stdout. It writes the alarm output to the local syslog.
smsManualRequester

**Purpose**
smsManualRequester sends update requests to the smsMaster.

**Configuration**
.smsManualRequester supports the following command-line options:

**Usage:**

```
smsManualRequester [-nodeid value]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>value</td>
<td>the ID of the node (Optional)</td>
</tr>
</tbody>
</table>

**Output**
smsManualRequester displays the output to local terminal.

**Examples:**

```
./smsManualRequester -nodeid 999
Nov 22 22:01:48 smsManualRequester(6578) NOTICE: Update Requester
`./smsManualRequester' process registered (node 999)
Enter table name (start with `-' to indicate delete)
?-to set info return:-ACS_CUSTOMER
Enter key names (key1,key2...):id
Enter values for keys & columns(terminate with ##):29
Enter values for keys & columns(terminate with ##):##
initialiseNode: Reading
`/IN/service_packages/SMS/etc/replication.def'
initialiseNode: heartbeatPeriod 20
initialiseNode: heartbeatTimeout 20
initialiseNode: connectionTimeout 2
initialiseNode: masterPortNum 12343
initialiseNode: queueWarnThresh 5
initialiseNode: queueErrThresh 100000
initialiseNode: queueCritThresh 1000000
initialiseNode: hBTolerance 10.0
initialiseNode: commitIdleTime 0.100000
initialiseNode: commitBusyTime 10.0
initialiseNode: tcpAbortSecs 20
initialiseNode: oracleUserPass '/'
initialiseNode: reportDir '/IN/service_packages/SMS/tmp/'
initialiseNode: statusFile '/IN/html/status.html'
initialiseNode: configFilePath
`/IN/service_packages/SMS/etc/replication.config'
initialiseNode: configFileName 'replication.config'
initialiseNode: node number 999
initialiseNode: node type 3
initialiseNode: s side updates 1
Nov 22 22:02:17 smsManualRequester(6578) NOTICE: Reached master
node 1 at `192.168.0.198'
Enter table name (start with `-' to indicate delete)
?-to set info return:
.....
```
**smsProcessCdr**

**Purpose**

smsProcessCdr processes EDRs based on the rules set in a format file. The format file describes the fields, literal strings and functions to apply to the input data in order to produce the desired output EDR.

Functions include:

- field selection
- reordering
- delimiter specification
- string concatenation with static strings and field values (such as would be required for field #13 in the EDR SRS), and
- multi-level pattern matching (as would be required for field #1) conditional field selection (as would be required for field #11).

This process is typically used to perform the following tasks for EDR files and ACS PIN log files:

1. (Optional) format conversion on files.
2. Move of files to a medium-term archive area.
3. (Optional) copy of files to directory for external retrieval.
4. Cleanup of expired files from the archive area.

Specification and implementation of EDR processing requirements is typically a system integration task which is performed prior to final system acceptance. This is usually implemented by the shell script smsCdrProcess.sh.

To prevent the EDR from being processed, see *Configuring smsCdrProcess.sh* (on page 105).

**Configuration**

smsProcessCdr accepts the following command line parameters.

**Usage:**

```bash
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t &lt;edr format&gt;</td>
<td>none</td>
<td>The filename of the EDR format file.</td>
</tr>
<tr>
<td>-d &lt;in dir&gt;</td>
<td>none</td>
<td>The directory to read EDRs from.</td>
</tr>
<tr>
<td>-D &lt;out dir&gt;</td>
<td>none</td>
<td>The directory to write processed EDRs to.</td>
</tr>
<tr>
<td>-s &lt;in suffix&gt;</td>
<td>none</td>
<td>The suffix that input EDR files must match (these are stripped from the output file name).</td>
</tr>
<tr>
<td>-p &lt;in prefix&gt;</td>
<td>none</td>
<td>The prefix that input EDR files must match (these are stripped from the output file name).</td>
</tr>
<tr>
<td>-S &lt;out suffix&gt;</td>
<td>none</td>
<td>The suffix to add to output EDR files.</td>
</tr>
</tbody>
</table>

*Note: No format conversion is performed by default. The formatting file supports the following: fields, literal strings, and functions.*

*Continued on next page*
**smsProcessCdr, Continued**

### Configuration (continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-P &lt;out prefix&gt;</td>
<td>none</td>
<td>The prefix to add to output EDR files.</td>
</tr>
<tr>
<td>-u &lt;usr/pwd&gt;</td>
<td>/</td>
<td>The oracle user and password to use.</td>
</tr>
<tr>
<td>-l &lt;tz&gt;</td>
<td>none</td>
<td>Alternate timezone TCS and TCE EDR fields are converted to.</td>
</tr>
<tr>
<td>-h</td>
<td>none</td>
<td>Displays a help page.</td>
</tr>
</tbody>
</table>

**Note:** smsProcessCdr will only attempt to connect to the database if the EDR format file contains functions that require it.

---

**Example 1**

The following command would:

- use `/IN/service_packages/SMS/bin/cdrFormat.fmt` as the EDR format file, and
- process every file matching the pattern
  `/IN/service_packages/SMS/cdr/received/scp2_acs*.cdr`

```
smsProcessCdr -t /IN/service_packages/SMS/bin/cdrFormat.fmt -d /IN/service_packages/SMS/cdr/received -D /tmp/processedCdrs -p scp2_acs -s .cdr -P ACS_ -S .out -u smf/smf
```

The output file name is a transformation of the input file name. For example, with the parameters supplied above, an input file

`/IN/service_packages/SMS/cdr/received/scp2_acs20010831120012.cdr`

would have output file name

`/tmp/processedCdrs/ACS_20010831120012.out`

---

**Example 2**

The following command:

```
smsProcessCdr -t /IN/service_packages/SMS/bin/mobistar.fmt -d /IN/service_packages/SMS/cdr/received -D /tmp/processedCdrs -p scp2_acs -s .cdr -P ACS_ -S .out -u smf/smf
```

would cause the following file to be parsed as the EDR format file:

`/IN/service_packages/SMS/bin/mobistar.fmt`

Once parsing is complete, the binary will process its input files. With the parameters supplied above, this would be every file matching the pattern:

`/IN/service_packages/SMS/cdr/received/scp2_acs*.cdr`

When an input EDR is successfully processed, it is written out to an output EDR file. One output EDR file is created for each input EDR file. The output file name is a transformation of the input file name.

This input file:

`/IN/service_packages/SMS/cdr/received/scp2_acs20010831120012.cdr`

would produce the following output file:

`/tmp/processedCdrs/ACS_20010831120012.out`

---

**Format File configuration**

A EDR format file consists of field specifiers which translate input data to an output format.

Continued on next page
The valid field specifiers are:

- **Standard Fields**
- **Special Fields**
- **Strings**
- **Functions**
- **Format Characters**

**Standard fields**

Standard fields are fields which relate to tags in the input EDR.

ACS EDRs have the following format:

```
<APPLICATION>|<tag1>=<value1>|<tag2>=<value2>| . . .
|<tagn>=<valuen>
```

A standard field is any one of the tag values.

If a EDR File format contains a field tag, then the corresponding value from the input EDR, value, will be written to the output EDR. Standard tags are written into the EDR format file using the same text which is used to specify them in the input EDR.

**Special fields**

Special fields are for data extracted from an EDR which does not occur in the input as a `<tag>=<value>` pair.

The only example of this - at this stage - is the EDR application name field, which always occurs as the first element in an EDR.

Placing the field `<APPLICATION>` in the EDR format file will cause the application name from the input EDR to be written to the output EDR.

**Strings**

Strings are used to write literal text to the output EDR.

Strings appear in the EDR format file as double quoted strings "<data>".

Any characters occurring in `<data>` are written, verbatim, to the output EDR file. This can be used to supply field delimiters (for example: ",") or to hard code output values (for example: "0,2,-1,").

**Functions**

Functions are programmatic transformations that can be applied to values.

Functions occur in the DR format file with the form:

```
<function name> ( <function parameters> )
```

Functions always produce a textual output.

The format of the functions used in smsProcessCdr are the same or similar to those used in the LISP programming language.

Most functions will support expressions as parameters so long as they produce textual output. The following types are included:

- standard fields
- special fields
- strings, and
- functions.

Continued on next page
smsProcessCdr, Continued

Functions (continued)

**Boolean expressions**

Boolean expressions are used as parameters to COND functions (and could be used as parameters to other functions at a later date).

Boolean functions are functions which evaluate to either TRUE or FALSE. The compiler will not allow Boolean functions to be used as 'top level' functions, but they may be nested in other functions to provide the ability to test conditions.

**EQUALS function**

The EQUALS function compares two expressions for equality. EQUALS evaluates to TRUE if the expressions are equivalent. Otherwise it evaluates to FALSE. The equality test is a case-sensitive string comparison.

```
EQUALS ( <expr1> , <expr2> )
```

**Example:** This example shows the EQUALS function being used as part of a COND function. EQUALS is used to check whether the application name is "ACS".

```
COND ( ( EQUALS ( <APPLICATION> , "ACS" ) , "ACS Service" ) , ( TRUE , "Unknown Service" ) )
```

**PREFIX function**

The PREFIX function evaluates to TRUE if expr2 is a prefix of expr1. Otherwise it evaluates to FALSE.

```
PREFIX ( <expr1> , <expr2> )
```

**Example:** This example shows the PREFIX function being used as part of a COND function. PREFIX is used to check whether the service number (SN) in the input EDR starts with either the digits 0800 or 0900.

```
COND ( ( PREFIX ( SN , "0800" ) , "Freephone" ) , ( PREFIX ( SN , "0900" ) , "Pay Service" ) , ( TRUE , "Unknown Service" ) )
```

**CONCAT function**

The CONCAT function concatenates one or more expressions.

```
CONCAT ( <expr1> [ , <expr2> , <expr3> . . . <exprn> ] )
```

**Example:** This example concatenates the literal string T0 onto the value of the special field, <APPLICATION>. Therefore, if <APPLICATION> evaluates to CCS then the example would produce the output: T0CCS.

```
CONCAT ( "T0" , <APPLICATION> )
```

**Example:** This example shows literal strings being concatenated to a field value and the result of another expression. If the value of the field XYZ is 21 and CCET is 12.32 then the result of the example would be: ABC2112.

```
CONCAT ( "ABC" , XYZ , ROUND ( CCET ) )
```

Continued on next page
Functions (continued)

**COND function**

The COND function evaluates to an expression on the basis of a series of one or more test Boolean expressions.

The first Boolean expression which evaluates to TRUE causes its associated result expression to be evaluated as the result of the COND function. If none of the Boolean expressions evaluates to TRUE then the result of the COND function is an empty string.

```cond
COND ( ( <bexpr1> , <expr1> ) [ , ( <bexpr2> , <expr2> ) . . . ,
( <bexprn> , <exprn> ) ] )
```

**Examples:**

In either of these two examples, the function evaluates to:

- Pizza Hut if the service number is 0800101101
- Pay Service if the service number starts with 0900, and
- Unknown in all other cases.

```cond
COND ( ( EQUALS ( SN , "0800101101" ) , "Pizza Hut" ) , ( TRUE , ( COND ( ( PREFIX ( SN , "0900" ) , "Pay Service" ) , ( TRUE , "Unknown" )
COND ( ( EQUALS ( SN , "0800101101" ) , "Pizza Hut" ) , ( PREFIX ( SN , "0900" ) , "Pay Service" ) , ( TRUE , "Unknown" )
```

For more examples, see the examples for PREFIX and EQUALS.

**LANGUAGEID function**

The LANGUAGEID function evaluates to the ID of a named language (from the ACS_SE_LANGUAGE table). The language name check is case insensitive.

If the named language can not be found, LANGUAGEID evaluates to –1.

```languageid
LANGUAGEID ( <string> )
```

**Note:** Using the LANGUAGEID function requires a connection to the database, which requires setting the oracle user / password option when invoking smsProcessCdr (unless the default “/” will suffice).

**Example:** This example checks the language ID from the input EDR (the LGID field). If the LGID is the same as the ID for the language English, then the expression evaluates to 1. If it is French, it evaluates to 2, if it is German, it evaluates to 3.

```cond
COND ( ( EQUALS ( LANGUAGEID ( "English" , LGID ) ) , "1" ) , ( EQUALS ( LANGUAGEID ( "French" , LGID ) ) , "2" ) , ( EQUALS ( LANGUAGEID ( "German" , LGID ) ) , "3" )
```
smsProcessCdr, Continued

Functions (continued)

ROUND function
The ROUND function interprets the supplied expression as a floating point number and replaces it with the same value rounded to the nearest integer. ROUND also works for negative numbers (using the minus symbol). If the supplied expression cannot be interpreted as a floating point number, then ROUND will evaluate to 0.

ROUND ( <expr> )

Examples:
This example evaluates to 2.
ROUND ( "2.1" )
This example evaluates to 3.
ROUND ( "2.6" )
If CCET evaluates to 12.3, this example evaluates to 12.
ROUND ( CCET )

SUBSTR function
The SUBSTR function extracts a substring from a given expression. The parameter <expr> is the source expression, <start> is an integer indicating where the substring should start (counting starts at zero) and <length> is an integer indicating how many characters should be read.

If <start> is greater than the length of <expr> then SUBSTR returns an empty string. If the specified <start> and <length> would cause SUBSTR to read off the end of the input expression, then SUBSTR returns the maximum number of characters it could read.

SUBSTR ( <expr> , <start> , <length> )

Examples:
This example evaluates to "the".
SUBSTR ( "the happy elephant" , 0, 3 )
This example evaluates to "e ha".
SUBSTR ( "the happy elephant" , 2, 4 )
This example evaluates to an empty string.
SUBSTR ( "the happy elephant" , 50, 4 )
This example evaluates to "appy elephant".
SUBSTR ( "the happy elephant" , 5, 40 )

Format characters
The format characters are a subset of the ASCII escape characters which allow special characters to be inserted into the output file. This table describes the supported format characters.

<table>
<thead>
<tr>
<th>Format character</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>\n</td>
<td>New line</td>
</tr>
<tr>
<td>\r</td>
<td>Carriage return</td>
</tr>
<tr>
<td>\t</td>
<td>Tab</td>
</tr>
</tbody>
</table>

Continued on next page
**smsProcessCdr**, Continued

### Format characters (continued)

<table>
<thead>
<tr>
<th>Format character</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>\0</td>
<td>Null</td>
</tr>
</tbody>
</table>

### File Format example 1

A simple example which just picks up the application name, service number, CLI.

Fields are comma delimited, records are terminated with a newline character (\n).

**Format File:**

```plaintext
<Application> "," SN "," CLI \n
```

**Input File Contents:**

```
ACS|SN=0800101101|CLI=044784333|XYZ=123
ACS|SN=0900222333|CLI=044784333|XYZ=123
ACS|SN=|CLI=044784333|XYZ=123

```

**Output File Contents:**

```
ACS,0800101101,044784333
ACS,0900222333,044784333
ACS,,044784333

```

### File Format example 2

A more complicated example using comments, special fields, and a function.

Fields are space delimited, records are terminated with a newline and a carriage return character.

**Format File:**

```plaintext
// our CDR format file
<Application> " "
// fields 2 and 3 are hard coded to be zero
"0 0 "
ROUND ( CCET ) " "
COND ( (EQUALS(<APPLICATION>, "CCS"), CONCAT("00", CA)), (TRUE, CONCAT("00", TN)))

```

**Input File Contents:**

```
CCS|XYZ=123|CCET=0.2|TN=123123|CA=321321|ABC=333
ACS|XYZ=123|CCET=8.8|TN=123123|CA=321321|ABC=333
VPN|XYZ=123|CCET=-1.6|TN=123123|CA=321321|ABC=333
CCS|XYZ=123|CCET=BOB|TN=123123|CA=321321|ABC=333

```

**Output File Contents:**

```
CCS 0 0 0 0321321
ACS 0 0 9 00123123
VPN 0 0 -2 00123123
CCS 0 0 0 0321321

```

### Further information

Because of the wide range of external EDR processing systems and site-specific requirements, it is not feasible in this document to describe all of the tasks which may be required to complete EDR integration.

For more information about this process, contact Level 1 support with details.
smsRecordStatistic

<table>
<thead>
<tr>
<th>Purpose</th>
<th>This tool makes use of the SMS statistics subsystem, which in turn makes use of shared memory for communicating with the smsStatsDaemon. The smsStatsDaemon must be installed and running.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location</td>
<td>The smsRecordStatistic process is located on the UAS in the following directory.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Configuration</th>
<th>smsRecordStatistic supports the following command-line options:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usage</td>
<td><strong>Usage:</strong></td>
</tr>
<tr>
<td></td>
<td>smsRecordStatistic [application] [statistic] [value]</td>
</tr>
<tr>
<td></td>
<td>The available parameters are:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[application]</td>
<td>The name of the application for the statistic. (Optional)</td>
<td></td>
</tr>
<tr>
<td>[statistic]</td>
<td>The name of the statistic to record. (Optional)</td>
<td></td>
</tr>
<tr>
<td>[value]</td>
<td>Adds the given delta value to the statistic. (Optional.)</td>
<td></td>
</tr>
</tbody>
</table>

| Output | The statistic named when running the script will be updated in the database. |
**startMerge**

**Purpose**
This command initiates a master merge of an Inferior Master to a Superior one. It can also be used to safely shut down a Superior Master by merging it with an Inferior Master.

**Configuration**
The startMerge supports the following command-line options:

**Usage:**
```
startMerge [-from <node num>] [-to <node num>]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-from &lt;node num&gt;</td>
<td>none</td>
<td>Node number of the inferior master to merge from.</td>
</tr>
<tr>
<td>-to &lt;node num&gt;</td>
<td>none</td>
<td>Node number of the inferior master to merge to.</td>
</tr>
</tbody>
</table>

**Failure**
If startMerge fails, it will write an error to the syslog and exit.

**Output**
The startMerge writes error messages to the system messages file, and also writes additional output to:
```
/IN/service_packages/SMS/tmp/merge.rep
```
# Reports

## Overview

**Introduction**

This chapter explains SMS reporting functionality.

**In this chapter**

This chapter contains the following topics.

- Reports Database Tables ................................................................. 190
- Installing a Report Script ............................................................... 192
- Report Script Worked Example ...................................................... 194
- Database Auditing ........................................................................... 199
Reports Database Tables

Introduction

Report-generating functionality is available via the SMS Java screens, to provide for service management reports of data.

This topic describes how to create reports, either:

- Service reports, to be installed at the time of service installation, or
- General reports, installed subsequently.

Database tables

There are three database tables which are specific to report generation:

- SMF_REPORT_SCRIPT contains one entry for each report script
- SMF_REPORT_PARAMETER contains one entry for each report parameter (may be none)
- SMF_REPORT_SCHEDULE contains one entry for each scheduled report instance. This table is not used for report installation, and is not covered in this document.

In addition, the following tables are used for controlling who has access to a report:

- SMF_APPLICATION
- SMF_APPLICATION_PART
- SMF_APPLICATION_ACCESS

These tables are reviewed here in regards to their role in report security. Security is handled by the standard SMS application part mechanism (see, example 3). Auditing is provided by the standard SMS audit mechanism, and should not need changing. The last change fields are the standard SMS last change fields, and are not listed in this table.

Report Scripts table

The report database table is called SMF_REPORT_SCRIPT. It contains the details of reports as shown below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORT_ID</td>
<td>Unique identifier, primary key, generated by a counter.</td>
</tr>
<tr>
<td>APP_ID</td>
<td>Application ID, foreign key to SMF_APPLICATION(app_id)</td>
</tr>
<tr>
<td>PART_ID</td>
<td>Part ID for security, foreign key to SMF_APPLICATION_PART (part_id)</td>
</tr>
<tr>
<td>CATEGORY</td>
<td>Script category, identifies the subdirectory.</td>
</tr>
<tr>
<td>SCRIPT</td>
<td>Script name, identifies the .sql file or shell script to execute.</td>
</tr>
<tr>
<td>NAME</td>
<td>Name to list in the report directory.</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Help text.</td>
</tr>
</tbody>
</table>

Report parameter table

The parameter table is called SMF_REPORT_PARAMETER. It contains the details of report parameters as shown below.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORT_ID</td>
<td>The ID of the report this parameter belongs to.</td>
</tr>
<tr>
<td>PARAM_NUMBER</td>
<td>The position of the parameter in the list, for example 1st, 2nd.</td>
</tr>
<tr>
<td>NAME</td>
<td>The parameter name.</td>
</tr>
</tbody>
</table>

Continued on next page
### Reports Database Tables, Continued

**Report parameter table (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESCRIPTION</td>
<td>Help text.</td>
</tr>
<tr>
<td>TYPE</td>
<td>The type – INT, STRING, DATE etc. (See table following for details)</td>
</tr>
<tr>
<td>DEFAULT_VALUE</td>
<td>Default value, optional.</td>
</tr>
<tr>
<td>VALID_VALUES</td>
<td>Valid comma separated values.</td>
</tr>
<tr>
<td>CONSTRAINT1</td>
<td>A constraint on the parameter (interpretation depends on TYPE).</td>
</tr>
<tr>
<td>CONSTRAINT2</td>
<td>A constraint on the parameter (interpretation depends on TYPE).</td>
</tr>
</tbody>
</table>
# Installing a Report Script

## Introduction

A script must be installed before it can be made available to the system. This process is described here, along with examples.

The main steps in the procedure detailed below are:

1. Choose an application ID, a category, and a report name.
2. Determine the parameters (if any) required by your script, and write the script.
3. Decide which application part your report will belong to.
4. Install the actual script on the USMS in the correct location.
5. Insert entries into the REPORT tables in the SMS database.

## Procedure

Follow these steps to install a report script.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | The Application ID must be an existing entry from SMF_APPLICATION. Common values are shown below. If you have additional services installed, additional choices may be available.  
   SQL> select app_id, application from smf_application;  
   APP_ID | APPLICATION  
   ------- | -----------  
   1       | SMS  
   4       | SYSTEM  
   2       | Acs_Service  
   The Category is an arbitrary name for a group of reports within one application. For example: "Customer", "Management", "Resource Usage".  
   The Name is a name for your report. Typically, this will be similar to the script name. For example, if your script is monthly_usage.sql, your report name could be "Monthly Usage".  
   Your script may take user parameters. The SMS report functions allow you to determine whether these are string, numeric, or values from a constrained list of parameters. Refer to the description of the SMF_REPORT_PARAMETER table to see the parameter types supported.  
   A report must either:  
   - Have a .sql extension. In this case, it will be executed using sqlplus.  
   or  
   - Be executable by the smf_oper user.  
   In either case, the script will be passed (n + 1) command line parameters, where n is the number of user parameters defined in SMF_REPORT_PARAMETER. Command line parameter one will always be the absolute output file name allocated to this report. |

Continued on next page
### Installing a Report Script, Continued

#### Procedure (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Examples:</strong></td>
<td></td>
</tr>
<tr>
<td>for a <code>.sql</code> file:</td>
<td><code>sqlplus &lt;script-name&gt; &lt;output-file&gt; [&lt;user-parameters&gt;]</code></td>
</tr>
<tr>
<td>for an executable file without a <code>.sql</code> extension</td>
<td><code>&lt;script-name&gt; &lt;output-file&gt; [&lt;user-parameters&gt;]</code></td>
</tr>
<tr>
<td>Exit status of report scripts are defined by the following:</td>
<td></td>
</tr>
<tr>
<td>0   = ok.</td>
<td></td>
</tr>
<tr>
<td>&gt;   = not ok. (Unix style).</td>
<td></td>
</tr>
<tr>
<td>&lt;0  = undefined.</td>
<td></td>
</tr>
<tr>
<td>Neither the smsReportsDaemon nor the smsReportScheduler is responsible for the clean up or reclaim of resources used by reports. This must be done explicitly by the application programmer.</td>
<td></td>
</tr>
<tr>
<td>The user may request cancellation of a script, in which case it will be sent a SIGTERM. Scripts should not ignore SIGTERM.</td>
<td></td>
</tr>
<tr>
<td>If the report spawns children, it should implement a SIGTERM handler to dispose the children, in case the user cancels a report.</td>
<td></td>
</tr>
</tbody>
</table>

4 For simplicity, an application part may be reused. Access to multiple reports may be controlled by one application part. You can even re-use access parts controlling existing installed screens.

You can list all existing defined application parts with the SQL command:

```sql
SQL> select app_id, part_id, part from smf_application_part;
```

If you choose to re-use application part 1030 (SMSReportScreens), all users who can access report screens will be able to run this report.

5 The script itself must be placed into:

```
/IN/service_packages/SMS/input/<application-name>/<category>/<script-file>
```

6 The script must now be made known to the SMS screens, and available for use to any SMS user who has access to the part ID, which owns your report script. This means:

- Inserting an entry into `SMF_REPORT_SCRIPT` for your script, indicating the category and script filename.
- Inserting one entry into `SMF_REPORT_PARAMETER` for each parameter in your report (if any). This indicates any constraints you wish enforced (for example, min/max values).
Report Script Worked Example

A script must be installed before it can be made available to the system. This process is described here, along with examples.

The main steps are:

1. Choose a category and a name for your script.
2. Determine the parameters (if any) required by your script, and write the script.
3. Decide which application part your report will belong to.
4. Install the actual script on the USMS in the correct location.
5. Insert entries into the REPORT tables in the SMF database.

Follow these steps to work through the example report script. The example script appears below description of each action in the procedure.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The details for this example are:</td>
</tr>
<tr>
<td></td>
<td>Application: SMS (ID is 1)</td>
</tr>
<tr>
<td></td>
<td>Category: “Errors”</td>
</tr>
<tr>
<td></td>
<td>Name: “Program Errors”</td>
</tr>
<tr>
<td></td>
<td>Script File: program_errors.sql</td>
</tr>
<tr>
<td></td>
<td>Application Part: 1805 (new part)</td>
</tr>
<tr>
<td></td>
<td>In this example, we are installing into the application SMS, which has the unique application ID of 1. Typically, you will install a service-specific report under the unique application ID that has been allocated to your service.</td>
</tr>
<tr>
<td></td>
<td>We are free to choose the category; we have chosen the “Errors” category.</td>
</tr>
<tr>
<td></td>
<td>We are free to choose a unique script name within this application and category; we have chosen &quot;program_errors.sql&quot;. We have chosen to create a new application part (1805) to control access to this report. An existing part may be used, for example: 1030.</td>
</tr>
</tbody>
</table>

2. The report takes three user parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num Hours</td>
<td>Integer in range 1..999, default is 24</td>
</tr>
<tr>
<td>Program Prefix</td>
<td>String length 0..20 characters</td>
</tr>
<tr>
<td>Category</td>
<td>One of FATAL, SERIOUS, WARNING, INFORMATIONAL</td>
</tr>
</tbody>
</table>

**Note:** The script itself will take four parameters. The first parameter is the output file name, which is determined by the reports daemon and is handed to us. You must generate your output to that file, if you wish it to be seen by the user.

The script below accepts three arguments, and shows the essential basic techniques of accepting input parameters, and spooling to the correct output file.

Continued on next page
Report Script Worked Example, Continued

**Example report script (continued)**

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>/*----------------------------------------------------------- */</td>
</tr>
<tr>
<td></td>
<td>* File: program_errors.sql</td>
</tr>
<tr>
<td></td>
<td>* Updates:</td>
</tr>
<tr>
<td></td>
<td>* Parameters: &amp;1 Output file, determined by reports daemon</td>
</tr>
<tr>
<td></td>
<td>* &amp;2 Hours back</td>
</tr>
<tr>
<td></td>
<td>* &amp;3 Program prefix</td>
</tr>
<tr>
<td></td>
<td>* &amp;4 Severity</td>
</tr>
<tr>
<td></td>
<td>* (FATAL, SERIOUS, WARNING, INFORMATIONAL)</td>
</tr>
<tr>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>* Copyright Notice:</td>
</tr>
<tr>
<td></td>
<td>* (c)1998 This source code is owned and copyrighted by G8</td>
</tr>
<tr>
<td></td>
<td><em>-----------------------------------------------------------------</em></td>
</tr>
<tr>
<td></td>
<td>-- #ident &quot;@(#)Id: telephony_errors.sql,v 1.4 1999/02/25 22:10:29 rhwang Exp &quot;$</td>
</tr>
<tr>
<td></td>
<td>-- so we won't print to stdout.</td>
</tr>
<tr>
<td></td>
<td>set termout off</td>
</tr>
<tr>
<td></td>
<td>set verify off</td>
</tr>
<tr>
<td></td>
<td>-- we are going to access the sms_program_errors table.</td>
</tr>
<tr>
<td></td>
<td>-- set the column titles.</td>
</tr>
<tr>
<td></td>
<td>column program 'Program' format a20</td>
</tr>
<tr>
<td></td>
<td>column error_code 'Error Code' format a16</td>
</tr>
<tr>
<td></td>
<td>column node_name 'Node Name' format a20</td>
</tr>
<tr>
<td></td>
<td>column severity 'Error Severity'</td>
</tr>
<tr>
<td></td>
<td>set linesize 80</td>
</tr>
<tr>
<td></td>
<td>set pagesize 2100</td>
</tr>
<tr>
<td></td>
<td>spool &amp;1</td>
</tr>
<tr>
<td></td>
<td>-- now set the title at the top of the page.</td>
</tr>
<tr>
<td></td>
<td>ttitle center 'Recent Telephony Errors for Application &amp;3 Severity &amp;4' skip 1 -</td>
</tr>
<tr>
<td></td>
<td>center 'PROGRAM TELEPHONY ERRORS' -</td>
</tr>
<tr>
<td></td>
<td>center ------------------------------------------ skip 1 -</td>
</tr>
<tr>
<td></td>
<td>RIGHT 'PAGE:' FORMAT 999 SQL.PNO SKIP 2</td>
</tr>
<tr>
<td></td>
<td>break on program skip 1;</td>
</tr>
<tr>
<td></td>
<td>break on severity skip 1;</td>
</tr>
<tr>
<td></td>
<td>select program, severity, error_code, node_name, timestamp</td>
</tr>
<tr>
<td></td>
<td>from sms_program_errors</td>
</tr>
<tr>
<td></td>
<td>where (program like '%&amp;3') and</td>
</tr>
<tr>
<td></td>
<td>(severity like '%&amp;4') and</td>
</tr>
<tr>
<td></td>
<td>(timestamp &lt; sysdate - &amp;2/24)</td>
</tr>
<tr>
<td></td>
<td>order by program, severity;</td>
</tr>
<tr>
<td></td>
<td>spool off</td>
</tr>
</tbody>
</table>
|      | quit

**Continued on next page**
Step | Action
--- | ---
3 | The user must now specify an application part for security purposes. If you have decided to re-use an existing part_id (for example: 1030), proceed to Example part 4.
1 | The Application ID for this example is 1. This is the unique ID for SMS.
2 | Application Part IDs must be in the range <App-ID> * 1000 + (0 ... 999) so for SMS, this means 1000 .. 1999. In this example, it has been determined that the ID 1805 is available for use. This is a new ID, which will control access to this report (and possibly others placed in the same security domain).
3 | Application Access IDs must be in the range <Part-ID> * 100 + (0..99). This is the part ID, so any Access ID within this range may be chosen. In the example, 180500 has been chosen.

As part of the installation for this script, run SQL to create this new Part ID.

**Note:** It is not necessary to create the SMF_APPLICATION for the USMS, since this is already created as part of the smsSms installation.

```sql
/*
 * Create our application part. We can re-use this if we 
 * have multiple reports that we want to have all controlled 
 * by a single security identifier
 */
insert into smf_application_part (part_id, app_id, part, description)
values (1805, 1, 'SMSErrorReports', 'Access SMS error reports category');
insert into smf_application_access(access_id, part_id, rights_name, description)
values (180500, 1805, 'Access', 'Run reports');

/*
 * We also add this to the 'SMS CreateDelete' user template, 
 * so that any user who is granted this template will get 
 * access to this report. We could add this to other 
 * templates too...
 */
var temp_id number;
EXEC select template_id into :temp_id from smf_template where template='SMS CreateDelete';
insert into smf_template_access (template_id, access_id)
values (:temp_id, 180500);

commit;

Continued on next page
Report Script Worked Example, Continued

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>As part of the installation package, ensure that the file, <code>program_errors.sql</code> is installed into the correct destination location, for example: <code>/IN/service_packages/SMS/input/SMS/Errors/program_errors.sql</code> The <code>smf_oper</code> user must have read access to this file. If this was a shell script or a binary program, it is necessary to ensure that the <code>smf_oper</code> also has execute access to this file.</td>
</tr>
<tr>
<td>5</td>
<td>The final task is to notify the USMS about the script, to make it visible. In this example, the report and the three user parameters to be collected are defined. <strong>Note:</strong> The screens constrain the content of the parameters to be passed to the script, but the interpretation of the parameters is of course up to the script itself.</td>
</tr>
</tbody>
</table>

```c
/*
 * Add our script to the list of scripts.
 */
var report_ref number;

insert into smf_report_script
(app_id, part_id, category, script, name, description)
values
(1, 1805, 'Errors', 'program_errors.sql', 'Program Errors', 'Dumps all the program errors for the specified program(s)');

exec select report_id into :report_ref -
from smf_report_script -
where (app_id=1) -
and (category = 'Errors') -
and (name = 'Program Errors');

insert into smf_report_parameter
(report_id, param_number, name, description, type, default_value, valid_values, constraint1, constraint2)
values
(:report_ref, 1, 'Num Days', 'Number of hours to go back', 'INT', '24', '', '1', '999');

insert into smf_report_parameter
(report_id, param_number, name, description, type, default_value, valid_values, constraint1, constraint2)
values
(:report_ref, 2, 'User', 'Leading string of program (0-20 characters)', 'STRING', '', '', '0', '20');
```

Continued on next page
### Example report script (continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><code>insert into smf_report_parameter (report_id, param_number, name, description, type, default_value, valid_values, constraint1, constraint2) values (:report_ref, 3, 'Category', 'Error Category (pulldown menu)', 'STRING', 'FATAL', 'FATAL,SERIOUS,WARNING,INFORMATIONAL', '', '');</code></td>
</tr>
<tr>
<td></td>
<td><code>commit;</code></td>
</tr>
</tbody>
</table>
### Database Auditing

#### Introduction
Changes to the data held in the SMF are tracked in the SMF_AUDIT table. The listAudit.sh tool enables reports to be run on the changes tracked in SMF_AUDIT.

#### Purpose
listAudit.sh enables you to run queries against the audit data held in the SMF_AUDIT table. The results are processes in to a comma separated report.

#### Configuration
listAudit.sh accepts the following command line options.

**Usage:**

```
listAudit.sh <usr/pwd> [<start_date>] [<end_date>] [<db user>] [<table>]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;usr/pwd&gt;</td>
<td></td>
<td>The user and password combination to be used to log into the SMF. (Required.)</td>
</tr>
<tr>
<td>&lt;start_date&gt;</td>
<td></td>
<td>The time and date the query will start reporting on. The format is yyyymmddhh24mmss. (Optional.)</td>
</tr>
<tr>
<td>&lt;end_date&gt;</td>
<td></td>
<td>The time and date the query will stop reporting on. The format is yyyymmddhh24mmss. (Optional.)</td>
</tr>
<tr>
<td>&lt;db user&gt;</td>
<td></td>
<td>The userid for the database user which made the changes to the database. (Optional.)</td>
</tr>
<tr>
<td>&lt;table&gt;</td>
<td></td>
<td>The database table which was changed. (Optional.)</td>
</tr>
</tbody>
</table>

The square brackets indicate optional parameters, but if a parameter is missed out and a later one used the missed out parameters should be indicated by using "".

#### Failure
If listAudit.sh fails, the report will not be completed. Errors will be sent to stderr.

#### Output
listAudit.sh writes error messages to the system messages file, and produces reports to stdout.

**Example:** This text shows an audit report for changes to the SMF_USER table by the SU user on the 08 Mar 2005.

```
$ listAudit.sh smf/smf 20050308000000 20050308235959 SU SMF_USER
Connected.
SU,20050308225724,192168007165,SMF_USER,ADMIN_TRAINING1_EX2,Student Training,Student 1,0,,31,20050321010942,LANGUAGE=ENGLISH,,,ADMIN_TRAINING1_EX2,Student Training,Student 1,0,Locked for testing,31,20050408000000,LANGUAGE=ENGLISH
SU,20050308225808,192168007165,SMF_USER,ADMIN_TRAINING1_EX1,Student Account,Student 1,0,,31,20050330023547,LANGUAGE=ENGLISH,,,ADMIN_TRAINING1_EX1,Student Account,Student 1,0,Locked for training,31,20050408000000,LANGUAGE=ENGLISH
SU,20050308225828,192168007165,SMF_USER,ADMIN_TRAINING1_EX1,Student Account,Student 1,0,Locked for training,31,20050408000000,LANGUAGE=ENGLISH,,,ADMIN_TRAINING1_EX1,Student Account,Student 1,0,,31,20050408000000,LANGUAGE=ENGLISH
```

Continued on next page
Database Auditing, Continued

Output (continued)

SU,20050308225838,192168007165,SMF_USER,ADMIN_TRAINING1_EX2,Student Training,Student 1,0,Locked for testing,31,20050408000000,LANGUAGE=ENGLISH
///,ADMIN_TRAINING1_EX2,Student Training,Student 1,0,31,20050408000000,LANGUAGE=ENGLISH
Troubleshooting

Overview

Introduction
This chapter explains common troubleshooting procedures and problem symptoms.

If the problem is not solved after consulting this chapter, contact Level 1 support. You will be asked to describe the failure in detail, together with any error messages that may have been displayed at the time of failure.

In this chapter
This chapter contains the following topics.

Common Troubleshooting Procedures .......................................................... 202
Possible Problems ....................................................................................... 205
Index defragmentation ................................................................................. 207
Common Troubleshooting Procedures

Introduction

This topic provides instructions for completing common troubleshooting procedures.

Checking current processes

You can check which processes are running using the standard UNIX command: ps. To find processes being run by Oracle software, you can grep for the string 'oper', which will display all processes being run by the application operator accounts (for example, acs_oper, ccs_oper and smf_oper).

Note: Some processes which are required for proper functioning may be run by other users, including root or the user which runs the webservers.

Example command: ps -ef | grep oper

For more information about the ps command, see the system documentation for the ps command.

You can also check how much of the processor a process is using by running the standard UNIX tool: top. If you have some baseline measurements, you will be able to compare it with the current load.

Example command: top

Tip: Some processes should only have one instance. If there are two or more instances, this may indicate a problem. For example, there will usually only be one timerIF running on each UAS.

For more information about which processes should be running on each node, check the Process List for each node in Installation.

Restarting running processes using kill

Follow these steps to restart a running process.

Important: Restarting some processes can cause system instability or data loss. Some processes must be restarted using specific tools. Check the documentation for the process before restarting.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Find the Process ID for the process you want to restart.  
Example command: ps -ef | grep smsAlarmRelay |
|      | Note: The second column of the results returned is the Process ID and the third column gives the Parent Process ID. |
| 2    | Kill the process using the kill command.  
Example command: kill -TERM 123  
Result: The process is terminated and will be restarted by the inittab process. |

Checking installed packages

To check the details of an installed package, use the pkginfo command.

Example command: pkginfo -l smsSms

Example output: This is an example of the output of the example command above.

PKGINST: smsSms  
NAME: Oracle smsSms  
CATEGORY: application  
ARCH: sun4u  
VERSION: 3.1.0  
VENDOR: Oracle  
PSTAMP: smsNode20041020104925  
INSTDATE: Oct 20 2004 13:15

Continued on next page
Checking installed packages (continued)

EMAIL: support@Oracle.com
STATUS: completely installed
FILES: 348 installed pathnames
        39 directories
        89 executables
        152448 blocks used (approx)

For more information about the pkginfo utility, see the system documentation.

Checking Oracle
A number of services and functions rely on access to the Oracle database. To check that Oracle is available to a service, check the following:

1. Use sqlplus to check that you can log into Oracle with the username and password the service is using to connect (these can usually be found in the service's configuration file).
   Example command: sqlplus smf/smf

2. Where the tables required for a service are known, use SQL queries to check that:
   - the tables exist, and
   - they have appropriate content.

For more information about SQL queries, see the Oracle documentation.

Checking network connectivity
Network connectivity will affect any process which requires communication between two different network addresses.

Network connectivity should support ssh sessions between the two machines experiencing the problem.

If you can open an ssh session between the two machines, check the following before contacting Level 1 support with details:

- If the address of either of the machines specified in the Node Management screens is a hostname, check that the hostnames used in the ssh sessions are the hostnames specified in the Node Management screen.

If you cannot ssh, check the following before contacting Level 1 support with details:

- Check that the hostname is resolving correctly in the DNS.
- Check that the physical network connection is working correctly.
- Check that the inetd and sshd are running.
- Check that sshd is listening on the expected port.
- Check that the smf_oper and acs_oper accounts are not locked, and that the username and password combinations being used are correct.
Common Troubleshooting Procedures, Continued

Replication

Replication may be failing for the following reasons:

- ssh keys have not been correctly set up between origin and destination machines.
- The destination node has been incorrectly set up in the Node Management screens of the SMS Java screens.
- Oracle is not running correctly.
- A new replication.cfg file has not been created after a change.
- replication.cfg may not be successfully copying to the destination machine (an error should display when the Create Config File button on the Node Management screens is clicked).
- The partition on the destination machine where the data is being replicated to may be full.
- The updateLoader on the destination machine may be running incorrectly.
- The destination database may be substantially out of sync with the SMF. Run a resync.

Checking configuration files

One of the significant areas where faults can occur and be remedied is in the configuration of processes. Configuration files can be edited by any standard text editor. A backup of the existing configuration file should always be taken before editing a configuration file.

For more information about the configuration files used in this application, see Configuration.

For more information about the configuration file for a specific program or tool, see the section named after the binary in question.
Possible Problems

This topic lists common problems and actions which can be taken to investigate or solve them. This list enables you to check for alarms based on the overall behaviour you are experiencing.

Follow these steps to resolve JavaClient problems.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ensure that the HTTPD daemon (on the USMS) is running and that it is correctly configured.</td>
</tr>
</tbody>
</table>
| 2    | If you are able to start the SMS screens, but unable to login:  
      | • Ensure that the sms.html file is correctly configured.  
      | • Ensure that the SMS console is able to resolve host names into IP addresses. |

This is caused by Java Runtime Environment (jre) running out of memory for the run time heap cache.

Under the default Java settings this may happen after 10 to 15 help screen accesses.

Follow these steps to extend the number of Help accesses.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Close the SMS screens.</td>
</tr>
<tr>
<td>2</td>
<td>From the Windows system, open the Control Panel.</td>
</tr>
<tr>
<td>3</td>
<td>Switch to Classic View to see the complete list of installed applications.</td>
</tr>
<tr>
<td>4</td>
<td>Double click Java icon to open java Control Panel.</td>
</tr>
<tr>
<td>5</td>
<td>Select the Java tab.</td>
</tr>
<tr>
<td>6</td>
<td>Click View in the Java Applet Runtime Settings panel.</td>
</tr>
<tr>
<td>7</td>
<td>Click the Java Runtime Parameter field.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is the fourth field along, pop-up may require expanding to see this field.</td>
</tr>
<tr>
<td>8</td>
<td>Type <code>-Xms10M -Xmx512M</code> in the Java Runtime Parameter field.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If other parameters are there, add these to the end.</td>
</tr>
<tr>
<td>9</td>
<td>Click OK.</td>
</tr>
<tr>
<td>10</td>
<td>Click Apply.</td>
</tr>
<tr>
<td>11</td>
<td>Click OK.</td>
</tr>
<tr>
<td>12</td>
<td>Close the Control Panel.</td>
</tr>
<tr>
<td>13</td>
<td>Restart the browser and start the SMS screen.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Using Xmx512M may cause issues with starting jre. If the browser jre cannot start up, try -Xmx180M.</td>
</tr>
</tbody>
</table>

Continued on next page
### Possible Problems, Continued

This table describes possible problems with replication.

<table>
<thead>
<tr>
<th>Alarm</th>
<th>Reason</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cannot connect to Oracle – exiting</td>
<td>There is a problem with the replication.config files in the system.</td>
<td>Use smsDumpRepConfig to check that the content of replication.config is correct. Generate a new replication.config file and check is it correctly copied to each machine. For more information, see replication.config File (on page 39).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Could not make fifo f – exiting</td>
<td>A connection is being dropped because the heartbeat settings on each end of a connection are different.</td>
<td>Check that the heartbeat settings for both ends of the connection are the same. The heartbeat settings are in replication.def, though they can be overridden at the command line for any process.</td>
</tr>
</tbody>
</table>

This table describes possible problems with comparisonServer.

<table>
<thead>
<tr>
<th>Alarm</th>
<th>Reason</th>
<th>Remedy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The replication.config file is not available to inetBootstrap, so smsCompareResync is not starting up.</td>
<td>Check that replication.config is in the correct directory and is readable by smf_oper.</td>
</tr>
</tbody>
</table>
Index defragmentation

<table>
<thead>
<tr>
<th>Description</th>
<th>The automatic defragmentation facility provided by SMS is intended to prevent fragmentation of the replication tables which frequently use insert, delete and update functions. In order to enable this defragmentation facility, the script <code>fragmentation_install.sh</code> must first be installed. This will install the stored procedure <code>sms_defrag_rep_iot</code>, and schedule a job to run it every 10 minutes. The following tables are affected:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>• REP_ORA_EVENT</td>
</tr>
<tr>
<td></td>
<td>• REP_ORA_RENUMBERED</td>
</tr>
</tbody>
</table>

Before you begin | The process for installing the defragmentation script varies depending on the Oracle configuration available on the SMP. For most clustered environments, Oracle configuration is stored in the service parameter file (SPFILE), which permits configuration parameters to be modified at runtime. If this is the case, then there is no need to manually alter the Oracle configuration. However, if SPFILEs are not in use (that is the traditional PFILEs are used to manage Oracle configuration), then it is important to first modify the cache and block sizings in the `initSMF.ora` file. The cache size for the 32K block size should be set to 32M or another suitably large value. |
|               | **Note:** It is recommended that this activity is performed by an experienced DBA. |

Enabling defragmentation | To enable the defragmentation facility, run the following script: `fragmentation_install.sh` This script is located in: `/IN/service_packages/SMS/db/defragmentation` |

Disabling defragmentation | To disable the defragmentation facility, run the uninstallation script as the oracle SMF user: `fragmentation_uninstall.sh` This script is located in: `/IN/service_packages/SMS/db/defragmentation` |

Oracle configuration restriction | While editing the parameter files, it must be noted that the following sets of parameters are mutually exclusive and cannot be used in combination with each other. Example: You cannot use one or more of: |
|                             | {db_cache_size, db_recycle_cache_size, db_keep_cache_size, db_nk_cache_size (where n is one of 2,4,8,16,32), db_cache_advice } |
|                             | AND one or more of the following in your configuration: |
|                             | {db_block_buffers} |

Continued on next page
Index defragmentation, Continued

Oracle configuration restriction (continued)

  buffer_pool_keep
  buffer_pool_recycle
Appendix

Overview

In this appendix

This appendix contains the following topics.

<table>
<thead>
<tr>
<th>Glossary of Terms</th>
<th>211</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index</td>
<td>215</td>
</tr>
</tbody>
</table>


# Glossary of Terms

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAA</td>
<td>Authentication, Authorisation, and Accounting. Specified in Diameter RFC 3588.</td>
</tr>
<tr>
<td>ACS</td>
<td>Advanced Control Services configuration platform.</td>
</tr>
<tr>
<td>ANI</td>
<td>Automatic Number Identification - Term used in the USA by long distance carriers for CLI.</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASN</td>
<td>Abstract Syntax Notation</td>
</tr>
<tr>
<td>CCS</td>
<td>1) Charging Control Services (or Prepaid Charging) component. 2) Common Channel Signalling. A signalling system used in telephone networks that separates signalling information from user data.</td>
</tr>
<tr>
<td>CLI</td>
<td>Calling Line Identification - the telephone number of the caller. Also referred to as ANI.</td>
</tr>
<tr>
<td>Connection</td>
<td>Transport level ink between two peers, providing for multiple sessions.</td>
</tr>
<tr>
<td>CORBA</td>
<td>Common Object Request Broker Architecture. It is a framework that provides interoperability between objects built in different programming languages, running on different physical machines perhaps on different networks. It specifies an Interface Definition Language, and API that allows client / server interaction with the ORB.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>cron</td>
<td>Unix utility for scheduling tasks.</td>
</tr>
<tr>
<td>crontab</td>
<td>File used by cron.</td>
</tr>
<tr>
<td>DB</td>
<td>Database</td>
</tr>
<tr>
<td>Diameter</td>
<td>A feature rich AAA protocol. Utilises SCTP and TCP transports.</td>
</tr>
<tr>
<td>DP</td>
<td>Detection Point</td>
</tr>
</tbody>
</table>
| EDR  | Event Detail Record  
**Note:** Previously CDR. The industry standard for CDR is EDR (Event Detail Record). Over time EDR will replace CDR in the Oracle documentation. |
| GUI  | Graphical User Interface |
| HTML | HyperText Markup Language, a small application of SGML used on the World Wide Web. It defines a very simple class of report-style documents, with section headings, paragraphs, lists, tables, and illustrations, with a few informational and presentational items, and some hypertext and multimedia. |
Hypertext Transport Protocol is the standard protocol for the carriage of data around the Internet.

Intelligent Network

Inter-operable Object Reference. A reference that is used in the CORBA world that clients can use to send their requests to a particular process executing on a particular machine. Every CORBA based server has an IOR that uniquely identifies it within a distributed computing platform. IOR consists of information such as the IP address of the machine on which the process is executing, or the port number to which it is listening. This IOR is usually exported/sent to some form of central registry when the process is started up. Clients can then retrieve this information, i.e. IORs, from the central registry if they want to send a request to a server.

1) Internet Protocol
2) Intelligent Peripheral - a box that is able to play announcements

Internet Protocol Address - network address of a card on a computer

The USMS Master Replicator is the Master Replicator which runs on the USMS. It is also know as a validator and always has the node number 1.

Measurement ID - used in Number Portability, counts the occurrences of an error.

Oracle Corporation

Object Request Broker. Within an Object based communication system, an ORB keeps track of the actual addresses of all defined objects and thus is used to route traffic to the correct destination. The CORBA defines the ORB in a series of standards enabling different platforms to share common information.

Provisioning Interface - used for bulk database updates/configuration instead of GUI based configuration.

Personal Identification Number

The concept of replication. This does not describe any single part of replication (i.e. ACSMaster of updateLoader), more the fact that information gets sent from one node to another.

Service Control Point. Also known as UAS.

Stream Control Transmission Protocol. A transport-layer protocol analogous to the TCP or User Datagram Protocol (UDP). SCTP provides some similar services as TCP (reliable, in-sequence transport of messages with congestion control) but adds high availability.


Service Logic Execution Environment

Service Logic Program Instance
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SMP</strong></td>
<td>Service Management Platform (also referred to as USMS).</td>
</tr>
<tr>
<td><strong>SMS</strong></td>
<td>Short Message Service.</td>
</tr>
<tr>
<td><strong>SN</strong></td>
<td>Service Number</td>
</tr>
<tr>
<td><strong>SNMP</strong></td>
<td>Simple Network Management Protocol. Usually responsible for notifying faults on a network.</td>
</tr>
<tr>
<td><strong>SQL</strong></td>
<td>Structured Query Language - a database query language.</td>
</tr>
<tr>
<td><strong>SSF</strong></td>
<td>Sub Service Field.</td>
</tr>
<tr>
<td><strong>SUA</strong></td>
<td>Signalling Connection Control Part User Adaptation Layer</td>
</tr>
<tr>
<td><strong>System Administrator</strong></td>
<td>The person(s) responsible for the overall set-up and maintenance of the IN.</td>
</tr>
<tr>
<td><strong>TCAP</strong></td>
<td>Transaction Capabilities Application Part – layer in protocol stack, message protocol.</td>
</tr>
<tr>
<td><strong>TCP</strong></td>
<td>Transmission Control Protocol. This is a reliable octet streaming protocol used by the majority of applications on the Internet. It provides a connection-oriented, full-duplex, point to point service between hosts.</td>
</tr>
<tr>
<td><strong>UAS</strong></td>
<td>Universal Application Server - hardware on which applications run.</td>
</tr>
<tr>
<td><strong>UBE</strong></td>
<td>Universal Billing Engine for Oracle Communications Network Control and Charging.</td>
</tr>
<tr>
<td><strong>USMS</strong></td>
<td>Universal Service Management System hardware platform.</td>
</tr>
<tr>
<td><strong>XML</strong></td>
<td>eXtensible Markup Language. It is designed to improve the functionality of the Web by providing more flexible and adaptable information identification. It is called extensible because it is not a fixed format like HTML. XML is a <code>metalanguage</code> — a language for describing other languages—which lets you design your own customized markup languages for limitless different types of documents. XML can do this because it's written in SGML.</td>
</tr>
</tbody>
</table>
Index

A
- a
  <days> • 134
  <dir> • 145
About this Document
  Audience • vii
  Changes in this document • vii
  Pre-requisites • vii
  Related documents • vii
  Scope • vii
Accessing Configuration Editor
  Screen and Access Permissions • 52
ACS • 2
address • 168
Alarm replication and buffering
  Alarms • 14
  smsAlarmDaemon • 94
Alarm Table fields
  SMF AlarmMessage Format • 77
Alarms • 5
  Alarm replication and buffering • 14
  Alarms diagram • 12
  Alarms replication process • 13
  Description of processes and executables • 13
  Enhanced Fault Management • 13
  Introduction • 12
  Statistics thresholds • 13
Alarms diagram
  Alarms • 12
Alarms replication process
  Alarms • 13
All nodes connected
  Replication in an Unclustered Installation • 29
All nodes isolated
  Replication in an Unclustered Installation • 32
ANI • 211
API • 19
Architectural overview
  smsTrigDaemon • 10
ASN • 75
Audience
  About this Document • vii
Auditing
  Auditing - listAudit.sh • 48
  Description • 48
Auditing - listAudit.sh
  Auditing • 48

B
Before you begin
  Index defragmentation • 207
  Boolean expressions • 182

C
- c • 120
  <dir> • 145
- C
  <secs> • 135
CCS • 2
CDR • 185
Changes in this document
  About this Document • vii
Checking configuration files
  Common Troubleshooting Procedures • 204
Checking current processes
  Common Troubleshooting Procedures • 202
Checking installed packages
  Common Troubleshooting Procedures • 202
Checking network connectivity
  Common Troubleshooting Procedures • 203
Checking Oracle
  Common Troubleshooting Procedures • 203
CLI • 10
cmnConfigRead
  Purpose • 92
  Startup • 92
cmnConfigSyntaxCheck
  Configuration • 152
  Output • 152
  Purpose • 152
cmnPushFiles • 19, 93
Clustered machine component diagram
  Machine Configuration • 7
cmnConfigRead
  Purpose • 92
  Startup • 92
cmnConfigSyntaxCheck
  Configuration • 152
  Output • 152
  Purpose • 152
cmnPushFiles • 19, 93
  Failure • 135
  Output • 135
  Purpose • 131
  Startup • 131
cmnReceiveFiles • 19
  Failure • 93
  Output • 93
  Parameters • 93
  Purpose • 93
  Startup • 93
C.,(continued)

Common Troubleshooting Procedures
Checking configuration files • 204
Checking current processes • 202
Checking installed packages • 202
Checking network connectivity • 203
Checking Oracle • 203
Introduction • 202
Restoring running processes using kill • 202

Components
community • 71
compareNode • 45
   Configuration • 153
   Failure • 153
   Output • 153
   Purpose • 153
comparisonServer is failing
Possible Problems • 206

Configuration component
smsTrigDaemon • 11
CONCAT function • 182
COND function • 183
configPlugin • 86
Configuration • 102, 119, 130, 179
cmnConfigSyntaxCheck • 152
cmnPushFiles • 131
compareNode • 153
Database Auditing • 199
infoDisplayer • 154
inputBootstrap • 155
repConfigWrite • 157
resyncServer • 158
smsAlarmDaemon • 94, 137
smsAlarmManager • 97
smsApplyConfig.sh • 130
smsCdrProcess.sh • 105
smsCompareResyncClient • 159
smsCompareResyncServer • 162
smsConfigDaemon • 102
smsConfigDaemonScript • 103
smsDumpRepConfig • 175
smsLogTest • 177
smsManualRequester • 178
smsMaster • 110
smsProcessCdr • 179
smsRecordStatistic • 186
smsSendConfig.sh • 119
startMerge • 187
Configuration component
Configuration Overview • 58
Configuration Management
What is the Service Management System? • 4

C.,(continued)

Configuration Management Editor
Editing XML Documents • 50
Introduction • 50
Schema Definition File • 50
Configuration Overview
Configuration components • 58
Configuration process overview • 58
Introduction • 58
Configuration process overview
Configuration Overview • 58

Configuring Nodes
Statistics nodes • 89
UAS Nodes • 89
USMS Nodes • 89

Configuring Replication Files
Example replication.def file • 64
Implementing change to replication.def • 64
Implementing changes to the replication.config file • 64
Introduction • 64
The replication.config file • 64
The replication.def file • 64

Configuring the Oracle Listener
Introduction • 66
Procedure • 66

Configuring the Resource Group in the Clustered Environment
Overview • 60
Starting the smsAlarmDaemon failover • 61
Starting the smsAlarmRelay failover • 61
Starting the smsNamingServer failover • 62
Starting the smsReportScheduler failover • 62
Starting the smsReportsDaemon failover • 63
Starting the smsStatsThreshold failover • 63
Starting the smsTaskAgent failover • 63
Starting the sshd failover • 60
Starting the webserver failover • 60

Configuring the SNMP Agent • 59, 99
Configuring the snmp.cfg file • 69
Formatting as a SNMP trap message • 74
Introduction • 69
Restarting the smsAlarmRelay • 76
SNMP relaying - switching on • 70
snmp.cfg example • 70
snmp.cfg file parameters • 70
Starting and stopping • 75
Transmission of the SNMP trap message • 75

Configuring the snmp.cfg file
Configuring the SNMP Agent • 69
Connection • 213
(continued)

Control plan execution requests
   smsTrigDaemon • 127
Conversion script • 54
   Distributing Configuration Files • 53
CORBA • 10
CORBA service port
   smsTaskAgent • 120
CPU • 77
cron • 48
crontab • 91
Customising the screens
   Example html applet • 87

Data flow
   What is Replication? • 22
   What is the Service Management System? • 4
Database Auditing
   Configuration • 199
   Failure • 199
   Introduction • 199
   Output • 199
   Purpose • 199
Database comparison components
   Database Comparisons • 45
Database comparison diagram
   Database Comparisons • 44
Database comparison process
   Database Comparisons • 45
Database Comparisons
   Database comparison components • 45
   Database comparison diagram • 44
   Database comparison process • 45
   Description • 44
Database resynchronisation components
   Database Resynchronisations • 47
Database resynchronisation diagram
   Database Resynchronisations • 46
Database resynchronisation process
   Database Resynchronisations • 47
Database Resynchronisations
   Database resynchronisation components • 47
   Database resynchronisation diagram • 46
   Database resynchronisation process • 47
   Description • 46

Database tables
   Reports Database Tables • 190
databaseHost • 84
databaseID • 84
DB • 68
Default.lang
   Defining the Screen Language • 79
Default.SMS.hs
   Defining the Help Screen Language • 81
Defining the Help Screen Language
   Default.SMS.hs • 81
   Example helpset language • 81
   Introduction • 81
   Procedure • 81
Defining the Screen Language
   Default.lang • 79
   Example screen language • 79
   Introduction • 79
   Procedure • 79
   User-specific language settings • 79
Description
   Auditing • 48
   Database Comparisons • 44
   Database Resynchronisations • 46
   Distributing Configuration Files • 53
   Index defragmentation • 207
   Replication Checks • 42
   smsStatsDaemon • 117
   What is the Service Management System? • 2
Description of processes and executables
   Alarms • 13
   EDRs • 19
   Statistics • 16
Description of resync processes and executables
   Replication in an Unclustered Installation • 33
Diameter • 51
Directory structure and filenames
   EDRs • 19
Disabling defragmentation
   Index defragmentation • 207
Distributing Configuration Files
   Conversion script • 53
   Description • 53
   Modifying eserv.config • 55
   Node identification • 53
   Process on Source node • 53
   Process on Target node • 54
   Reloading configuration data • 55
Document Conventions
   Icons • viii
   Typographical conventions • viii
D. (continued)

Dual Network Connection
  Maintaining Network Connections • 9

E

-e • 134
  <secs> • 141

Editing XML Documents
  Configuration Management Editor • 50

EDR • 19

EDR file transfer diagram
  EDRs • 18

EDR file transfer process
  EDRs • 18

EDR format
  smsCdrProcess.sh • 105

EDR intermediate file format
  EDRs • 19

EDRs
  Description of processes and executables • 19
  Directory structure and filenames • 19
  EDR file transfer diagram • 18
  EDR file transfer process • 18
  EDR intermediate file format • 19
  Introduction • 18

Enabling defragmentation
  Index defragmentation • 207

Enhanced Fault Management
  Alarms • 13
  Example • 135
  Example 1
    smsProcessCdr • 180
  Example 2
    smsProcessCdr • 180
  Example helpset language
    Defining the Help Screen Language • 81
  Example html applet • 85, 86
    Customising the screens • 87
  Example replication.config
    replication.config File • 39
  Example replication.def file
    Configuring Replication Files • 64
    replication.def File • 37
  Example report script
    Report Script Worked Example • 194
  Example screen language
    Defining the Screen Language • 79
    extraFIFO • 125

F

-f
  <dir/file> • 142
  <dir> • 133
  -F • 134, 143
  -f <dir> • 131

Failover
  Replication in an Unclustered Installation • 28

Failover and Error Recovery
  Further information • 25
  Introduction • 25
  Update queuing • 25
  updateLoader failure • 25

Failure
  cmnPushFiles • 135
  cmnReceiveFiles • 93
  compareNode • 153
  Database Auditing • 199
  infMaster • 136
  inputBootstrap • 156
  repConfigWrite • 157
  smsAlarmDaemon • 96, 138
  smsAlarmManager • 98
  smsAlarmRelay • 100
  smsApplyConfig.sh • 130
  smsCdrProcess.sh • 105
  smsConfigDaemon • 102
  smsConfigDaemonScript • 104
  smsDbCleanup.sh • 106
  smsDumpRepConfig • 175
  smsLogCleaner • 107, 139
  smsLogTest • 177
  smsMaster • 110
  smsMergeDaemon • 109
  smsNamingServer • 111
  smsReportCleanupStartup.sh • 116
  smsReportScheduler • 115
  smsReportsDaemon • 113
  smsSendConfig.sh • 119
  smsStatsDaemon • 145
  smsStatsThreshold • 118
  smsTaskAgent • 122
  smsTrigDaemon • 126
  startMerge • 187
  updateLoader • 150

File Format example 1
  smsProcessCdr • 185

File Format example 2
  smsProcessCdr • 185

Format characters
  smsProcessCdr • 184

Format File configuration
  smsProcessCdr • 180
F (continued)

Formatting as a SNMP trap message
Configuring the SNMP Agent • 74
Functions
  smsProcessCdr • 181
  What is the Service Management System? • 2
Further information
  Failover and Error Recovery • 25
  replication.config File • 40
  smsProcessCdr • 185

G

-g • 122
Generating replication.config
  replication.config File • 39
  group • 171
  Groups • 170
  groups-cover-table-on-scp • 169
  GUI • 212

H

-h
  <host> • 132
  <ratio> • 142
Heartbeating
  Maintaining Network Connections • 9
High Level Architecture
  SMS - Xerlin Integration • 51
HTML • viii
HTTP • 11

I

-i • 143
  <dir> • 114
  <host> • 120
Icons
  Document Conventions • viii
  id • 167
Implementing change to replication.def
  Configuring Replication Files • 64
Implementing changes to the replication.config file
  Configuring Replication Files • 64
IN • vii
Index defragmentation
  Before you begin • 207
  Description • 207
  Disabling defragmentation • 207
  Enabling defragmentation • 207
  Oracle configuration restriction • 207
Inferior Master nodes
  Replication in an Unclustered Installation • 27

I (continued)

infMaster • 26
  Failure • 136
  Output • 136
  Parameters • 136
  Purpose • 136
  Startup • 136
infoDisplayer
  Configuration • 154
  Output • 154
  Purpose • 154
Input file
  smsCompareResyncServer • 164
Input file example
  smsCompareResyncServer • 173
inputBootstrap • 45, 47
  Configuration • 155
  Failure • 156
  Purpose • 155
Installing a Report Script
  Introduction • 192
  Procedure • 192
Installing Additional Applications
  Installing the applications • 90
  Order of replication • 90
Installing the applications
  Installing Additional Applications • 90
Interactive reports
  smsReportsDaemon • 113
Introduction
  Alarms • 12
  Common Troubleshooting Procedures • 202
  Configuration Management Editor • 50
  Configuration Overview • 58
  Configuring Replication Files • 64
  Configuring the Oracle Listener • 66
  Configuring the SNMP Agent • 69
  Database Auditing • 199
  Defining the Help Screen Language • 81
  Defining the Screen Language • 79
  EDRs • 18
  Failover and Error Recovery • 25
  Installing a Report Script • 192
  Maintaining Network Connections • 9
  Possible Problems • 205
  replication.config File • 39
  replication.def File • 35
  Report Script Worked Example • 194
  Reports Database Tables • 190
  Setting up the screens • 83
  SMF AlarmMessage Format • 77
  smsNamingServer • 111
Statistics • 15
What is Replication? • 22
I (continued)
IOR • 111
IP • 48
IP address • 48
IPC • 67
Isolated UAS
Replication in an Unclustered Installation • 30
Isolated USMS
Replication in an Unclustered Installation • 31

J
Java help screen greyed out
Possible Problems • 205

K
key-columns • 169

L
LANGUAGEID function • 183
Legacy mode configuration - config file
smsStatsDaemon • 146
listenPort • 71, 74, 124
Location
smsRecordStatistic • 186
smsTrigDaemon • 123
logjob.conf • 59
smsLogCleaner • 108, 140
logo • 85

M
-m
<size> • 143
-M
<secs> • 135
-m <size> • 143, 144
Machine Configuration
Clustered machine component diagram • 7
Overview • 6
Single machine component diagram • 8
Single machine configuration • 7
Unclustered machine configuration • 6
Maintaining Network Connections
Dual Network Connection • 9
Heartbeating • 9
Introduction • 9
Master Controllers
What is Replication? • 24
Master Replicator • 9
masterConfigFile • 87
-maxpending
<int> • 110

M (continued)
Measurement IDs - standard mode
smsStatsDaemon • 145
Measurements • 148
Merging nodes
Replication in an Unclustered Installation • 33
Message flows
smsTrigDaemon • 10
MIB field mappings - SMF_ALARM_DEFN
SMF AlarmMessage Format • 78
MIB field mappings - SMF_ALARM_MESSAGE
SMF AlarmMessage Format • 77
MID • 146
Modifying eserv.config • 54, 130
Distributing Configuration Files • 55
my-addr • 72
my-oid • 73

N
namingServerPort • 85
Node • 167
Node identification
Distributing Configuration Files • 53
Node numbers
Replication in an Unclustered Installation • 28
nodes • 172
Nodes
What is Replication? • 22
notification-oid • 74
NP • 147

O
-o
<dir> • 114, 133
-o <dir> • 134
opaque • 72, 73
Oracle • ii
Oracle configuration restriction
Index defragmentation • 207
oracleLogin • 124
ORB • 88
Order of replication
Installing Additional Applications • 90
other-columns • 170
Output
cmnConfigSyntaxCheck • 152
cmnPushFiles • 135
cmnReceiveFiles • 93
compareNode • 153
Database Auditing • 199
infMaster • 136
infoDisplayer • 154
repConfigWrite • 157
### Overview
- Configuring the Resource Group in the Clustered Environment • 60
- Machine Configuration • 6

#### Parameters
- `-p` 100
- `<port>` 111, 121, 132
- `-P`
- `<dir>` 131
- `-a <days>` 134
- `-a <dir>` 145
- `address` 168
- `-c` 120
- `-c <dir>` 145
- `-C <secs>` 135
- `-clusterDatabaseHost` 85
- `-cmnReceiveFiles` 93
- `-community` 71
- `-configPlugin` 86
- `-d <dir>` 131
- `-d <rows>` 142
- `-databaseHost` 84
- `-databaseID` 84
- `-e` 134
- `-e <secs>` 141
- `extraFIFO` 125
- `-F` 134, 143
- `-f <dir/file>` 142
- `-f <dir>` 133
- `-g` 122
- `group` 171
- `Groups` 170
- `groups-cover-table-on-scp` 169
- `-h <host>` 132
- `-h <ratio>` 142
- `-i` 143
- `-i <dir>` 114
- `-i <host>` 120
- `id` 167
- `infMaster` 136
- `key-columns` 169
- `listenPort` 71, 124
- `logo` 85
- `-M <secs>` 135
- `-m <size>` 143
- `masterConfigFile` 87
- `maxpending <int>` 110
- `my-addr` 72
- `my-oid` 73
- `namingServerPort` 85
- `Node` 167
- `nodes` 172
- `notification-oid` 74
- `-o <dir>` 114, 133
- `opaque` 73
- `oracleLogin` 124
- `other-columns` 170
- `-p` 100
- `-P <dir>` 131
- `-p <port>` 111, 121, 132
- `-param-oid` 73
- `perform` 165
- `port` 73
- `produce-final-reports` 165
- `-r <node>` 144
- `-r <pref>` 132
- `-R <secs>` 134
- `ranges` 172
- `Replication` 164
- `report-after` 166
- `report-directory` 166
- `report-row-number-limit` 165
- `-S` 143
- `-s <Kb>` 145
- `-s <port>` 121
- `-s <secs>` 99, 118, 132
- `-S <sufx>` 131
- `scps` 126
- `smsAlarmRelay` 99
- `smsDbCleanup.sh` 106

#### Command Line Options
- `O, (continued)
  - resyncServer • 158
  - smsAlarmDaemon • 96, 138
  - smsAlarmManager • 98
  - smsAlarmRelay • 101
  - smsApplyConfig.sh • 130
  - smsCdrProcess.sh • 105
  - smsCompareResyncServer • 173
  - smsConfigDaemon • 102
  - smsConfigDaemonScript • 104
  - smsDbCleanup.sh • 106
  - smsDumpRepConfig • 175
  - smsLogCleaner • 107, 139
  - smsLogTest • 177
  - sms<ManualRequester> • 178
  - smsMaster • 110
  - smsMergeDaemon • 109
  - smsNamingServer • 111
  - smsRecordStatistic • 186
  - smsReportCleanupStartup.sh • 116
  - smsReportScheduler • 115
  - smsReportsDaemon • 113
  - smsSendConfig.sh • 119
  - smsStatsDaemon • 145
  - smsStatsThreshold • 118
  - smsTaskAgent • 122
  - smsTrigDaemon • 126
  - startMerge • 187
  - updateLoader • 150

- `P, (continued)
  - extraFIFO • 125
  - `-F` 134, 143
  - `-f <dir/file>` 142
  - `-f <dir>` 133
  - `-g` 122
  - `group` 171
  - `Groups` 170
  - `groups-cover-table-on-scp` 169
  - `-h <host>` 132
  - `-h <ratio>` 142
  - `-i` 143
  - `-i <dir>` 114
  - `-i <host>` 120
  - `id` 167
  - `infMaster` 136
  - `key-columns` 169
  - `listenPort` 71, 124
  - `logo` 85
  - `-M <secs>` 135
  - `-m <size>` 143
  - `masterConfigFile` 87
  - `maxpending <int>` 110
  - `my-addr` 72
  - `my-oid` 73
  - `namingServerPort` 85
  - `Node` 167
  - `nodes` 172
  - `notification-oid` 74
  - `-o <dir>` 114, 133
  - `opaque` 73
  - `oracleLogin` 124
  - `other-columns` 170
  - `-p` 100
  - `-P <dir>` 131
  - `-p <port>` 111, 121, 132
  - `-param-oid` 73
  - `perform` 165
  - `port` 73
  - `produce-final-reports` 165
  - `-r <node>` 144
  - `-r <pref>` 132
  - `-R <secs>` 134
  - `ranges` 172
  - `Replication` 164
  - `report-after` 166
  - `report-directory` 166
  - `report-row-number-limit` 165
  - `-S` 143
  - `-s <Kb>` 145
  - `-s <port>` 121
  - `-s <secs>` 99, 118, 132
  - `-S <sufx>` 131
  - `scps` 126
  - `smsAlarmRelay` 99
  - `smsDbCleanup.sh` 106

- `P, (continued)
  - extraFIFO • 125
  - `-F` 134, 143
  - `-f <dir/file>` 142
  - `-f <dir>` 133
  - `-g` 122
  - `group` 171
  - `Groups` 170
  - `groups-cover-table-on-scp` 169
  - `-h <host>` 132
  - `-h <ratio>` 142
  - `-i` 143
  - `-i <dir>` 114
  - `-i <host>` 120
  - `id` 167
  - `infMaster` 136
  - `key-columns` 169
  - `listenPort` 71, 124
  - `logo` 85
  - `-M <secs>` 135
  - `-m <size>` 143
  - `masterConfigFile` 87
  - `maxpending <int>` 110
  - `my-addr` 72
  - `my-oid` 73
  - `namingServerPort` 85
  - `Node` 167
  - `nodes` 172
  - `notification-oid` 74
  - `-o <dir>` 114, 133
  - `opaque` 73
  - `oracleLogin` 124
  - `other-columns` 170
  - `-p` 100
  - `-P <dir>` 131
  - `-p <port>` 111, 121, 132
  - `-param-oid` 73
  - `perform` 165
  - `port` 73
  - `produce-final-reports` 165
  - `-r <node>` 144
  - `-r <pref>` 132
  - `-R <secs>` 134
  - `ranges` 172
  - `Replication` 164
  - `report-after` 166
  - `report-directory` 166
  - `report-row-number-limit` 165
  - `-S` 143
  - `-s <Kb>` 145
  - `-s <port>` 121
  - `-s <secs>` 99, 118, 132
  - `-S <sufx>` 131
  - `scps` 126
  - `smsAlarmRelay` 99
  - `smsDbCleanup.sh` 106
Pre-requisites

About this Document • vii
Primary replication nodes
What is Replication? • 23
Procedure
Configuring the Oracle Listener • 66
Defining the Help Screen Language • 81
Defining the Screen Language • 79
Installing a Report Script • 192
Process Descriptions
What is the Service Management System? • 4
Process on Source node
Distributing Configuration Files • 53
Process on Target node
Distributing Configuration Files • 54
produce-final-reports • 165
Purpose
cmnConfigRead • 92
cmnConfigSyntaxCheck • 152
cmnPushFiles • 131
cmnReceiveFiles • 93
cmpareNode • 153
Database Auditing • 199
infMaster • 136
infoDisplayer • 154
inputBootstrap • 155
repConfigWrite • 157
resyncServer • 158
smsAlarmDaemon • 94, 137
smsAlarmManager • 97
smsAlarmRelay • 99
smsApplyConfig.sh • 130
smsCdrProcess.sh • 105
smsCompareResyncClient • 159
smsCompareResyncServer • 162
smsConfigDaemon • 102
smsConfigDaemonScript • 103
smsDbCleanup.sh • 106
smsDumpRepConfig • 175
smsLogCleaner • 107, 139
smsLogTest • 177
smsManualRequester • 178
smsMaster • 110
smsMergeDaemon • 109
smsProcessCdr • 179
smsRecordStatistic • 186
smsReportCleanupStartup.sh • 116
smsReportScheduler • 114
smsReportsDaemon • 112
smsSendConfig.sh • 119
smsStatsDaemon • 144
smsStatsThreshold • 118
smsTaskAgent • 120
smsTrigDaemon • 10, 123

Parameters - replication.def
replication.def File • 35

Parameters for legacy mode • 144
smsStatsDaemon • 144

Parameters for standard mode • 144
smsStatsDaemon • 144

param-oid • 73, 74
perform • 165

Pl • 10
smsTrigDaemon • 11

PIN • 131
port • 73

Possible Problems
comparisonServer is failing • 206
Java help screen greyed out • 205
Replication is failing • 206
SMS Java screens will not start • 205

P,(continued)
Related documents

- About this Document • vii
- Reloading configuration data • 54
- Distributing Configuration Files • 55

repConfigWrite

- Configuration • 157
- Failure • 157
- Output • 157
- Purpose • 157
- Startup • 157

Replication • 164

- Common Troubleshooting Procedures • 204
- What is the Service Management System? • 4

Replication Check • 25

Replication check components

- Replication Checks • 42

Replication check diagram

- Replication Checks • 42

Replication check process

- Replication Checks • 43

Replication Checks

- Description • 42
- Replication check components • 42
- Replication check diagram • 42
- Replication check process • 43

Replication component diagram

- Replication in an Unclustered Installation • 26

Replication Components

- Replication in an Unclustered Installation • 26

Replication groups

- What is Replication? • 23

Replication in an Unclustered Installation

- All nodes connected • 29
- All nodes isolated • 32
- Description of resync processes and executables • 33
- Failover • 28
- Inferior Master nodes • 27
- Isolated UAS • 30
- Isolated USMS • 31
- Merging nodes • 33

Replication is failing

- Possible Problems • 206
- Replication Overview • 4
- Replication process

- What is Replication? • 22

replication.config File • 40, 59, 157, 175, 206
- Example replication.config • 39
- Further information • 40
- Generating replication.config • 39
- Introduction • 39

replication.def File • 58, 109
- Example replication.def file • 37
- Introduction • 35
- Parameters - replication.def • 35

Replicator • 9

Report parameter table

- Reports Database Tables • 190

Report Script Worked Example

- Example report script • 194
- Introduction • 194

Report Scripts table

- Reports Database Tables • 190

report-after • 166

report-directory • 166

Reporting Functions

- What is the Service Management System? • 4

report-row-number-limit • 165, 166

Reports Database Tables

- Database tables • 190
- Introduction • 190
- Report parameter table • 190
- Report Scripts table • 190

Resend Alarms

- smsAlarmRelay • 100

Restarting running processes using kill

- Common Troubleshooting Procedures • 202

Restarting the smsAlarmRelay

- Configuring the SNMP Agent • 76

resyncServer • 46, 47

- Configuration • 158
- Output • 158
- Purpose • 158

ROUND function • 184
S

<s> • 145
</s>

<port> • 121
<secs> • 99, 118, 132
-S • 143
<sufx> • 131

Schema Definition File
  Configuration Management Editor • 50

Scope
  About this Document • vii
SCP • 104
scps • 126
Screen and Access Permissions
  Accessing Configuration Editor • 52
SCTP • 211
Setting up the screens
  Introduction • 83
SGML • 211
Single machine component diagram
  Machine Configuration • 8
Single machine configuration
  Machine Configuration • 7
SLEE • 11
SLPI • 146
SMF AlarmMessage Format
  Alarm Table fields • 77
  Introduction • 77
  MIB field mappings - SMF_ALARM_DEFN
    • 78
  MIB field mappings - SMF_ALARM_MESSAGE • 77
  SMF Listen Messages • 78
SMF Listen Messages
  SMF AlarmMessage Format • 78
SMP • 5
SMS • vii
SMS - Xerlin Integration
  High Level Architecture • 51
  Xerlin License Requirements • 51
SMS component diagram
  What is the Service Management System?
    • 3
SMS Java screens will not start
  Possible Problems • 205
SMS subsystems
  What is the Service Management System?
    • 3
smsAlarmDaemon • 13, 14, 61
  Alarm replication and buffering • 94
    Configuration • 94, 137
    Failure • 96, 138
    Output • 96, 138
    Purpose • 94, 137
    Startup • 94, 137
  Usage examples • 95, 138

S (continued)

smsAlarmManager • 14
  Configuration • 97
  Failure • 98
  Output • 98
  Purpose • 97
  Startup • 97
smsAlarmRelay • 61, 76
  Failure • 100
  Output • 101
  Parameters • 99
  Purpose • 99
  Resend Alarms • 100
  Startup • 99
smsApplyConfig.sh
  Configuration • 130
  Failure • 130
  Output • 130
  Purpose • 130
  Startup • 130
smsCdrProcess.sh • 19
  Configuration • 105
  EDR format • 105
  Failure • 105
  Output • 105
  Purpose • 105
  Startup • 105
smsCompareResyncClient • 34, 43, 45, 47
  Configuration • 159
  Purpose • 159
smsCompareResyncServer • 33, 43
  Configuration • 162
  Input file • 164
  Input file example • 173
  Output • 173
  Purpose • 162
smsConfigDaemon • 103
  Configuration • 102
  Failure • 102
  Output • 102
  Purpose • 102
  Startup • 102
smsConfigDaemonScript
  Configuration • 103
  Failure • 104
  Output • 104
  Purpose • 103
  Startup • 103
smsDbCleanup.sh
  Failure • 106
  Output • 106
  Parameters • 106
  Purpose • 106
  Startup • 106
S (continued)

smsDumpRepConfig • 40, 157
  Configuration • 175
  Failure • 175
  Output • 175
  Purpose • 175
smsLogCleaner
  Failure • 107, 139
  logjob.conf • 108, 140
  Output • 107, 139
  Parameters • 107, 139
  Purpose • 107, 139
  Startup • 107, 139
smsLogTest
  Configuration • 177
  Failure • 177
  Output • 177
  Purpose • 177
smsManualRequester
  Configuration • 178
  Output • 178
  Purpose • 178
smsMaster • 4, 13, 14, 16, 26, 33, 45, 143
  Configuration • 110
  Failure • 110
  Output • 110
  Purpose • 110
  Startup • 110
smsMergeDaemon • 27
  Failure • 109
  Output • 109
  Parameters • 109
  Purpose • 109
  Startup • 109
smsNamingServer • 27, 62
  Failure • 111
  Introduction • 111
  Output • 111
  Parameters • 111
  Startup • 111
smsProcessCdr • 105
  Configuration • 179
  Example 1 • 180
  Example 2 • 180
  File Format example 1 • 185
  File Format example 2 • 185
  Format characters • 184
  Format File configuration • 180
  Functions • 181
  Further information • 185
  Purpose • 179
  Special fields • 181
  Standard fields • 181
  Strings • 181
S (continued)

smsRecordStatistic
  Configuration • 186
  Location • 186
  Output • 186
  Purpose • 186
smsReportCleanupStartup.sh
  Failure • 116
  Output • 116
  Parameters • 116
  Purpose • 116
  Startup • 116
smsReportScheduler • 62
  Failure • 115
  Output • 115
  Parameters • 114
  Purpose • 114
  Startup • 114
Unix utilities • 115
smsReportsDaemon • 5, 14, 16, 19, 63
  Failure • 113
  Interactive reports • 113
  Output • 113
  Parameters • 112
  Purpose • 112
  Startup • 112
smsSendConfig.sh
  Configuration • 119
  Failure • 119
  Output • 119
  Purpose • 119
  Startup • 119
smsStatsDaemon • 16, 117
  Description • 117
  Failure • 145
  Legacy mode configuration - config file • 146
  Measurement IDs - standard mode • 145
  Output • 145
  Parameters • 141
  Parameters for legacy mode • 144
  Parameters for standard mode • 144
  Purpose • 141
  smsStatsDaemon configuration • 141
  smsStatsDaemon parameters • 146
  Startup • 141
  Statistics shared memory • 146
  Updating smsStatsDaemon measurements • 149
  smsStatsDaemon configuration
  smsStatsDaemon • 141
  smsStatsDaemon parameters
  smsStatsDaemon • 146
smsStatsThreshold • 14, 16, 63
  Failure • 118
  Output • 118
  Parameters • 118
  Purpose • 118
  Startup • 118
smsTaskAgent • 5, 27, 63
  CORBA service port • 120
  Failure • 122
  Output • 122
  Parameters • 120
  Purpose • 120
  Startup • 120
smsTrigDaemon
  Architectural overview • 10
  Components • 11
  Control plan execution requests • 127
  Data consistency check • 127
  Failure • 126
  Location • 123
  Message flows • 10
  Output • 126
  Parameters • 123
  PI • 11
  Purpose • 10, 123
  Startup • 123
  xmlTcapInterface • 11
SN • 20
SNMP • 14
SNMP relaying - switching on
  Configuring the SNMP Agent • 70
SNMP v1 • 74
SNMP v3 • 75
snmp.cfg example
  Configuring the SNMP Agent • 70
snmp.cfg file parameters
  Configuring the SNMP Agent • 70
Special fields
  smsProcessCdr • 181
specific • 72, 73
SQL • 106
SSF • 17
Standard fields
  smsProcessCdr • 181
Starting and stopping
  Configuring the SNMP Agent • 75
Starting the smsAlarmDaemon failover
  Configuring the Resource Group in the
  Clustered Environment • 61
Starting the smsAlarmRelay failover
  Configuring the Resource Group in the
  Clustered Environment • 61
StartMerge • 33, 109
  Configuration • 187
  Failure • 187
  Output • 187
  Purpose • 187
Startup
  cmnConfigRead • 92
  cmnPushFiles • 131
  cmnReceiveFiles • 93
  infMaster • 136
  repConfigWrite • 157
  smsAlarmDaemon • 94, 137
  smsAlarmManager • 97
  smsAlarmRelay • 99
  smsApplyConfig.sh • 130
  smsCdrProcess.sh • 105
  smsConfigDaemon • 102
  smsConfigDaemonScript • 103
  smsDbCleanup.sh • 106
  smsLogCleaner • 107, 139
  smsMaster • 110
  smsMergeDaemon • 109
  smsNamingServer • 111
  smsReportCleanupStartup.sh • 116
  smsReportScheduler • 114
  smsReportsDaemon • 112
  smsSendConfig.sh • 119
  smsStatsDaemon • 141
  smsStatsThreshold • 118
  smsTaskAgent • 120
  smsTrigDaemon • 123
  updateLoader • 150
Statistics • 5
  Description of processes and executables • 16
  Introduction • 15
  Statistics collected • 17
  Statistics collection diagram • 15
  Statistics collection process • 16
  Statistics thresholds • 16
Statistics collected
  Statistics • 17
Statistics collection diagram
  Statistics • 15
Statistics collection process
  Statistics • 16
Statistics nodes
  Configuring Nodes • 89
Statistics shared memory
  smsStatsDaemon • 146
Statistics thresholds
  Alarms • 13
  Statistics • 16
stop-on-limit • 166
Strings
  smsProcessCdr • 181
SUA • 137
SUBSTR function • 184
Superior Master nodes
  What is Replication? • 23
Symbolic names
  <bits> • 132
  <days> • 134
  <dir/file> • 142
  <dir> • 114, 131, 133, 145
  <host> • 120, 121, 132
  <int> • 110
  <Kb> • 145
  <node> • 144
  <port> • 111, 121, 132
  <pref> • 132
  <ratio> • 142
  <rows> • 142
  <secs> • 99, 118, 121, 132, 133, 134, 141, 145
  <size> • 143
  <sufx> • 131
  <usr/pwd> • 99, 111, 114, 118, 121, 144
Syntax for the stats_config file • 147
System Administrator • 44
System Overview • 91
U, (continued)

Usage examples
  smsAlarmDaemon • 95, 138
useFIFO • 125
useORB • 124, 125
userName • 71
User-specific language settings
  Defining the Screen Language • 79
use-SNMPv3 • 70, 71, 74
USMS • vii
USMS Nodes
  Configuring Nodes • 89

V
  -v • 115, 142
  variable bindings • 75
  view • 167
  VPN • 147

W
  -w • 144
  <secs> • 121, 133
  webdavDebug • 86
  webdavPort • 86
What is Replication? • 150
  Data flow • 22
  Introduction • 22
  Master Controllers • 24
  Nodes • 22
  Primary replication nodes • 23
  Replication groups • 23
  Replication process • 22
  Superior Master nodes • 23
  Update Loader nodes • 23
  Update Requester nodes • 23
  Update requests to primary nodes • 24
What is the Service Management System?
  Configuration Management • 4
  Data flow • 4
  Description • 2
  Functions • 2
  Process Descriptions • 4
  Replication • 4
  Reporting Functions • 4
  SMS component diagram • 3
  SMS subsystems • 3
  workingConfigFile • 87

X
  -x • 100, 133
Xerlin License Requirements
  SMS - Xerlin Integration • 51
XML • vii
xmlTcapInterface
  smsTrigDaemon • 11