Oracle® Communications
Network Charging and Control
Service Management System Technical Guide
Release 5.0.1

June 2013
Copyright

Copyright © 2013, 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 is software or related documentation that 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 END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware 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 that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware 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

About This Document ........................................................................................................ vii
Document Conventions ..................................................................................................... viii

Chapter 1
System Overview .................................................................................................................. 1

Overview ............................................................................................................................. 1
What is the Service Management System? ........................................................................ 1
Platform Configuration ....................................................................................................... 4
Maintaining Network Connections ..................................................................................... 6
smsTrigDaemon .................................................................................................................. 7
Alarms ................................................................................................................................. 9
Statistics ............................................................................................................................. 12
EDRs ................................................................................................................................ 15

Chapter 2
Replication Overview ......................................................................................................... 19

Overview ............................................................................................................................ 19
What is Replication? .......................................................................................................... 19
Failover and Error Recovery ............................................................................................. 22
Replication in an Unclustered Installation ....................................................................... 23
replication.def File .......................................................................................................... 30
replication.config File .................................................................................................... 37

Chapter 3
Replication Check ............................................................................................................. 41

Overview ............................................................................................................................ 41
Replication Checks ............................................................................................................ 41
Database Comparisons ..................................................................................................... 43
Database Resynchronizations .......................................................................................... 45
Auditing ............................................................................................................................. 47

Chapter 4
Configuring the Environment .......................................................................................... 49

Overview ............................................................................................................................ 49
Configuration Overview ..................................................................................................... 49
Configuring the Resource Group in the Clustered Environment ...................................... 51
Configuring Replication Files .......................................................................................... 54
Configuring the Oracle Listener ....................................................................................... 55
Configuring the SNMP Agent ......................................................................................... 58
SMF AlarmMessage Format ............................................................................................. 64
Defining the Screen Language ......................................................................................... 66
Defining the Help Screen Language ................................................................................ 68
Setting up the Screens ..................................................................................................... 69
Configuring Nodes .......................................................................................................... 73
Installing Additional Applications .................................................................................... 73
### Chapter 5

**Background Processes on the SMS** .......................................................... 75

<table>
<thead>
<tr>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmnConfigRead</td>
</tr>
<tr>
<td>cmnReceiveFiles</td>
</tr>
<tr>
<td>smsAlarmDaemon</td>
</tr>
<tr>
<td>smsAlarmManager</td>
</tr>
<tr>
<td>smsAlarmRelay</td>
</tr>
<tr>
<td>smsConfigDaemon</td>
</tr>
<tr>
<td>smsConfigDaemonScript</td>
</tr>
<tr>
<td>smsCdrProcess.sh</td>
</tr>
<tr>
<td>smsDbCleanup.sh</td>
</tr>
<tr>
<td>smsLogCleaner</td>
</tr>
<tr>
<td>smsMergeDaemon</td>
</tr>
<tr>
<td>smsMaster</td>
</tr>
<tr>
<td>smsNamingServer</td>
</tr>
<tr>
<td>smsReportsDaemon</td>
</tr>
<tr>
<td>smsReportScheduler</td>
</tr>
<tr>
<td>smsReportCleanupStartup.sh</td>
</tr>
<tr>
<td>smsStatsDaemon</td>
</tr>
<tr>
<td>smsStatisticsWriter</td>
</tr>
<tr>
<td>smsStatsThreshold</td>
</tr>
<tr>
<td>smsSendConfig.sh</td>
</tr>
<tr>
<td>smsTaskAgent</td>
</tr>
<tr>
<td>smsTrigDaemon</td>
</tr>
</tbody>
</table>

### Chapter 6

**Background Processes on the SLC** ..................................................... 117

<table>
<thead>
<tr>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmsApplyConfig.sh</td>
</tr>
<tr>
<td>cmnPushFiles</td>
</tr>
<tr>
<td>infMaster</td>
</tr>
<tr>
<td>smsAlarmDaemon</td>
</tr>
<tr>
<td>smsLogCleaner</td>
</tr>
<tr>
<td>smsStatsDaemon</td>
</tr>
<tr>
<td>updateLoader</td>
</tr>
</tbody>
</table>

### Chapter 7

**Tools and Utilities** ........................................................................ 139

<table>
<thead>
<tr>
<th>Process</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmnConfigSyntaxCheck</td>
</tr>
<tr>
<td>cmnSU</td>
</tr>
<tr>
<td>compareNode</td>
</tr>
<tr>
<td>comparisonServer</td>
</tr>
<tr>
<td>inetCompareServer</td>
</tr>
<tr>
<td>infoDisplayer</td>
</tr>
<tr>
<td>inputBootstrap</td>
</tr>
<tr>
<td>repConfigWrite</td>
</tr>
<tr>
<td>resyncServer</td>
</tr>
<tr>
<td>smsCompareResyncClient</td>
</tr>
<tr>
<td>smsCompareResyncServer</td>
</tr>
<tr>
<td>smsDumpRepConfig</td>
</tr>
</tbody>
</table>
About This Document

Scope

The scope of this document includes all the information required to install, configure and administer the Service Management System 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.

Prerequisites

Although there are no prerequisites 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 prerequisite 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/)
Document Conventions

Typographical Conventions

The following terms and typographical conventions are used in the Oracle Communications Network Charging and Control (NCC) documentation.

<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 names of tabs.</td>
</tr>
<tr>
<td></td>
<td>Names of database tables and fields.</td>
</tr>
<tr>
<td><em>Italics</em></td>
<td>Name of a document, chapter, topic or other publication.</td>
</tr>
<tr>
<td></td>
<td>Emphasis within text.</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>Example: To close the window, either click <strong>Close</strong>, or press <strong>Esc</strong>.</td>
<td></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>Example: Ctrl+P, or Alt+F4.</td>
<td></td>
</tr>
</tbody>
</table>

**Monospace**
Examples of code or standard output.

**Monospace Bold**
Text that you must enter.

**variable**
Used to indicate variables or text that should be replaced.

**menu option > menu option >**
Used to indicate the cascading menu option to be selected, or the location path of a file.
Example: Operator Functions > Report Functions
Example: /IN/html/SMS/Helptext/

**hypertext link**
Used to indicate a hypertext link on an HTML page.

Specialized terms and acronyms are defined in the Glossary at the end of this guide.
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 specific Oracle Communications Network Charging and Control (NCC) network or service implications of the product.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the Service Management System?</td>
<td>1</td>
</tr>
<tr>
<td>Platform Configuration</td>
<td>4</td>
</tr>
<tr>
<td>Maintaining Network Connections</td>
<td>6</td>
</tr>
<tr>
<td>smsTrigDaemon</td>
<td>7</td>
</tr>
<tr>
<td>Alarms</td>
<td>9</td>
</tr>
<tr>
<td>Statistics</td>
<td>12</td>
</tr>
<tr>
<td>EDRs</td>
<td>15</td>
</tr>
</tbody>
</table>

What is the Service Management System?

Description

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

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

SMS provides the following:

- A central repository for other IN services, such as ACS and CCS
- 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
Chapter 1

- Auditing of Database Changes

**SMS component diagram**

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

**SMS subsystems**

There are four main subsystems within SMS:

- Replication
- Configuration management
- Reporting functions
• File transfer

**Replication**

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 SLCs into the SMF
  • Changes to persistent data held in the SMF to all relevant SLCs (so all parts of the system have consistent data)
• Alternative network routing between the SLC and SMS under network failure, or buffered updates under complete network failure or SMS downtime
• 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 VWS)
• Any update of application data due to the actions of the service running on the SLCs (including client account and call routing data)
• System, application and interface statistics
• Alarms

For more information about replication, see *Replication Overview* (on page 19).

**Reporting Functions**

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.

**Data flow**

There are two main methods of data transfer:

• Replication
• File transfer (using ftp)

**Process Descriptions**

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><em>smsMaster</em> (on page 90)</td>
</tr>
<tr>
<td>SMF</td>
<td>The main SMF on the SMS.</td>
<td></td>
</tr>
<tr>
<td>SCP</td>
<td>The databases on the SLCs. They hold a subset of the data on the SMF.</td>
<td></td>
</tr>
<tr>
<td>updateLoaders</td>
<td>Receives update orders from the smsMaster and inserts them into the SCPs.</td>
<td><em>updateLoader</em> (on page 136)</td>
</tr>
</tbody>
</table>
Platform Configuration

Overview

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

- On a single platform
- With one SMS on one platform and one or more SLCs on separate platforms
- With multiple SMSs connected to a RAID array and one or more SLCs on separate platforms

Unclustered platform configuration

Using the unclustered platform configuration, the smsSms package is installed on the SMS. The smsScp package is installed on one or more SLCs.

This configuration provides resilience by using a failover system from the SMS to the SLCs. However, while the SMS is unavailable, no configuration updates can be forwarded to the SLCs.
Unclustered platform configuration

Here is an example of replication in an unclustered installation.

Single platform configuration

Using the single platform configuration, all required SMS functionality is installed on a single platform. Because all SMS functionality is on a single computer 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.
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 acknowledgment. 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)
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 favor 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 SMS 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 SLC platform. An indication of whether or not the requests were successful passes back from the ACS to the initiating client.
Architectural overview

This diagram shows smsTrigDaemon and components that surround it.

Message flows

This table describes the 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 SLC 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 SLC 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>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 SLC platform.</td>
</tr>
<tr>
<td>5</td>
<td>The xmlTcapInterface constructs an InitialDP and sends it to ACS through the SLEE. For more information about the SLEE, see SLEE Technical Guide.</td>
</tr>
<tr>
<td>6</td>
<td>An indication of success or failure is returned to the xmlTcapInterface using a Connect, Continue or ReleaseCall component.</td>
</tr>
</tbody>
</table>
Chapter 1

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
- 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 User's & 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 SMS and SLCs 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
- 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.

Alarms replication 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 SMS and SLCs. Sources include:  
|       | • The syslog file  
|       | • Oracle logs |
| 2     | The smsAlarmDaemon sends an update request to the superior master (usually the smsMaster).  
|       | **Exception:** The smsAlarmDaemon on the SMS makes its updates directly to the SMF,
When the superior master receives an update request, it will insert the updated data into the SMF_ALARMS_MESSAGE table of the SMF.

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.

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).

### Statistics thresholds

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 *SMS User's Guide*.

### Enhanced Fault Management

Enhanced Fault Management (EFM) 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.

### 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>smsAlarmDaemon</td>
<td>Collects alarms from local sources and forwards them to the smsMaster.</td>
<td>smsAlarmDaemon (on page 77)</td>
</tr>
<tr>
<td>smsMaster</td>
<td>Receives alarms from smsAlarmDaemons and forwards them to the SMF.</td>
<td>smsMaster (on page 90)</td>
</tr>
<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 90)</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 92)</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 107)</td>
</tr>
</tbody>
</table>
### Alarm replication and buffering

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

- Protection against the SMS being flooded with alarms
- Filtering of repeating alarms

For more information about buffering alarms, see *smsAlarmDaemon* (on page 77).

### Statistics

#### Introduction

Statistics generated by the SMS and SLCs 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
- Running reports against the statistics held in SMF_STATISTICS

For more information about using these functions, see *SMS User's Guide*.
Statistics collection diagram

Here is an example of the statistics collection process.

Description of processes and executables

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 SLCs and forwards them to smsMaster.</td>
<td>smsStatsDaemon (on page 128).</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 90).</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 92).</td>
</tr>
<tr>
<td>smsStatsThreshold</td>
<td>Monitors the SMF_STATISTICS table in the SMF. If the statistics meet certain rules, the smsStatsThreshold process</td>
<td>smsStatsThreshold (on page 107).</td>
</tr>
</tbody>
</table>
Further information will create an alarm and forward it to the smsAlarmDaemon on the SMS.

### Statistics collection process

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>
</table>
| 1     | Statistics are gathered by the statistics daemon process (smsStatsDaemon) which runs on each SLC platform. Statistics which are collected include:  
- Statistics from the shared memory which are generated by the slee_acs  
- TCAP statistics from files saved by the TCAP interface  
- System statistics from the kernel |
| 2     | At regular intervals, the smsStatsDaemon sends the values to the smsMaster process on the SMS platform as an update request. |
| 3     | The smsMaster adds the new statistics to the SMF_STATISTICS table in the SMF. |
| 4     | The administrator can run reports on the collected statistics using the statistics screens in SMS (which are executed by the smsReportsDaemon). |

### Statistics thresholds

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 *SMS User's Guide*.

### Statistics collected

The statistics system can collect any SMS-compatible IN application statistics. These are typically coarse values related to the general performance and behavior 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 one partners

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 *ACS Technical Guide*. 

---

14  NCC Service Management System Technical Guide
EDRs

Introduction

The SMS software provides a complete, integrated reporting mechanism for Event Detail Records (EDR). It allows the developers of SMS-compatible IN applications to add report functions to their product, through the SMS reports interface.

EDR file transfer diagram

Here is an example of the transfer of files between SLCs and the SMS.

EDR file transfer process

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 SLCs, cmnPushFiles collects files from the configured input directory and transfers them to the configured output directory on the SMS 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>Stage</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</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 SMS, 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>

**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 using stdout. Depending on the success of the transfer, the file will also be moved to another directory on the origination SLC.</td>
<td>cmnPushFiles (on page 118).</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 76).</td>
</tr>
<tr>
<td>smsCdrProcess.sh</td>
<td>Provides a set of EDR processing and archiving functions.</td>
<td>smsCdrProcess.sh (on page 86).</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 92).</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 `ACS Technical Guide`.

The file name is `ApplicationID.cdr`. In this case, the complete specification of the currently active EDR filename for the ACS application will be `APP_yyyymmddhhmmss.txt`.

Where:
- **APP** is the three letter acronym for the originating process
- **yyyymmddhhmmss** 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 SLC and SMS 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 `/IN/service_packages/SMS/cdr/current/` directory.

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:**
Chapter 1

# File created at 1999060312449
Acs_Service|SN=1800906420|TN=4770360|CGN=9380360|TCS=1999060312449
Acs_Service|SN=1800906421|TN=4770361|CGN=9380361|TCS=1999060312450
Acs_Service|SN=1800906422|TN=4770362|CGN=9380362|TCS=1999060312457
Acs_Service|SN=1800906423|TN=4770363|CGN=9380363|TCS=1999060312521
Acs_Service|SN=1800906424|TN=4770364|CGN=9380364|TCS=1999060312547
Acs_Service|SN=1800906425|TN=4770365|CGN=9380365|TCS=1999060312449
Acs_Service|SN=1800906426|TN=4770366|CGN=9380366|TCS=1999060312449
Acs_Service|SN=1800906427|TN=4770367|CGN=9380367|TCS=1999060313036
Acs_Service|SN=1800906428|TN=4770368|CGN=9380368|TCS=1999060312036
Overview

Introduction

This chapter explains the replication system used in SMS.

In this chapter

This chapter contains the following topics.

What is Replication?  
Failover and Error Recovery  
Replication in an Unclustered Installation  
replication.def File  
replication.config File

What is Replication?

Introduction

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

Data flow

The SMF database on the SMS holds the full set of authoritative data within the system. Data required for call processing and resilience is forwarded to the SCP database on the SLCs using SMS replication. Updates are received from processes on the SMSs and the SLCs and from the Service Management System administration screens.
Replication process

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 one of the following:  
   - The administration screens  
   - An event on the SMS or SLCs |
| 2     | If the update comes from the administration screens, it will be one of:  
   - Forwarded to the smsTaskAgent, and then through to an smsMaster  
   - Be inserted directly into the SMF database  
   If the update request comes from the SMS or SLCs, 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)  
   2. Spawn a local smsMaster (child) process to insert the updated data into the SMF database. |
| 4     | The updateLoader on the relevant SLCs will read the update order from the socket and insert the data into the SCP 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 SMS User's Guide.

Superior Master nodes

Superior master nodes are forwarded all data update requests within SMS, and distribute update orders to all SLCs that require the replication data through the updateLoaders.

In a clustered installation, the superior master role is shared between the available smsMaster nodes on the SMSs.

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 SMS, but at times may be an infMaster on an SLC.

Update Loader nodes

Update loader nodes will run on any SLC that requires database updates. They are the updateLoader processes running on the SLCs. They accept update orders from superior master nodes and insert the data into the local SCP database.

The update loaders on a single SLC 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.

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.
Update Requester nodes

Update requesters create update requests in response to specific events on the SLCs and send them to the superior master to update the centralized data (and from there it will be replicated out to the relevant SLCs). Update requesters include:

- replicationIF
- smsAlarmsDaemons
- smsStatsDaemons

Update requesters do not need to be configured in the database.

Replication groups

A replication table has one or more replication groups. A replication group can be assigned to one or more replication nodes.

Example:

- Replication Group A resides on Node 1, Node 2 and Node 3
- Replication Group B resides on Node 1 and Node 3

Primary replication nodes

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.

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.

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.

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:
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-synchronization 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 acknowledgment 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 a file named using the following format:

`updateLoaderNodeNumber-queuedOrders.dat`

#### Further Information

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

Replication component diagram

Here is an example of replication in an unclustered installation.

Replication components

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 SMS handling updates throughout SMS. This is the superior master for all connected nodes.</td>
<td>smsMaster (on page 90)</td>
</tr>
<tr>
<td>infMaster</td>
<td>An infMaster runs on each SLC. 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 123)</td>
</tr>
</tbody>
</table>
updateLoaders | An updateLoader runs on each SLC. It manages all incoming update orders and inserts updated data into the SCP. At any point in time, an updateLoader will be connected to a specific superior master. | updateLoader (on page 136)

update requesters | Update requesters may run on any machine. They send update requests to the Superior Master. | Update Requester nodes (on page 21)

smsMergeDaemon | The smsMergeDaemon runs on the SMS and monitors the connections between the SMS and the SLCs. If it notices a break in the connection, it may start a merge to update the disconnected nodes. | smsMergeDaemon (on page 89)

smsTaskAgent | 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 SLCs. | smsTaskAgent (on page 109)

smsNamingServer | The smsNamingServer enables non-SMS components to connect to elements within the SMS. | smsNamingServer (on page 91)

SMF | This Oracle database holds authoritative data for all SLCs. | 

SCPs | These Oracle databases hold the subset of SMF data required to route calls. | 

**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).

**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 assigned to the smsMaster process on the SMS.</td>
</tr>
<tr>
<td>17-255</td>
<td>These node numbers are available to infMaster processes on the SLCs.</td>
</tr>
<tr>
<td>256-511</td>
<td>These node numbers are available to updateLoaders on the SLCs. Usually they start from node number 301.</td>
</tr>
</tbody>
</table>
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

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 SMS), 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 SLCs 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.
All nodes connected

Here is an example showing all nodes in an unclustered configuration connected to the smsMergeDaemon.
Isolated SLC

This diagram depicts an isolated SLC in an unclustered environment.

Where an SLC 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, SLC1 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 SLC1) and connects to that.

The Master 1 queues all updates for SLC1 until such time as it comes back on line. When SLC1 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 SLC1.

If the smsMergeDaemon is not running, the startMerge process may be used instead. startMerge will copy the data from SLC1 to the smsMaster. The smsMaster will then update the rest of the network with the information received from SLC1.
Isolated SMS

This diagram depicts an isolated SMS.

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.

In the above case the Master 1 on the SMS 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 SLC1 and SLC2 will find and connect to inferior master 2 on SLC1. When the SMS comes back into the network, the smsMergeDaemon will check each SLC infMaster process to see if there are any connections to them. In this case, there are connections to the SLC1 infMaster process (node 2) from the SLC2 (node 3). So the smsMergeDaemon will run startMerge against SLC1. startMerge will copy SLC1’s data across to the SMS. The smsMaster will then attempt to update both SLCs with the new data from SLC1.
All nodes isolated

This diagram depicts all nodes isolated.

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 SLC1 connecting to inferior master 2 on SLC1 (node 2), and the update loader on SLC2 connecting to the inferior master 3 (node 3).

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

Merging nodes

If a infMaster is acting as a superior master, it will collect update requests in a table on the local SCP. 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.
Chapter 2

The process for completing this task is known as a merge. Usually, the smsMergeDaemon will initiate a merge automatically when the connection has stabilized. 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 171).

**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. If the smsMaster cannot connect to a updateLoader, it will collect pending updates until a new connection to the updateLoader is made.</td>
<td>smsMaster (on page 90)</td>
</tr>
<tr>
<td>resyncServer</td>
<td>Takes a snapshot of the SMF and sends it to the compareResyncReceive process on the SLC. 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 149)</td>
</tr>
<tr>
<td>compareResyncReceiver</td>
<td>Updates the SCP with the data from the SMF (sent by resyncServer on the SMS).</td>
<td></td>
</tr>
<tr>
<td>smsCompareResyncClient</td>
<td>Receives information from smsCompareResyncServer and updates the SCP.</td>
<td>smsCompareResyncClient (on page 147)</td>
</tr>
<tr>
<td>updateLoader</td>
<td>When a resync is started, the updateLoader will stop making updates to the SCP. Instead it writes the updates to a file named in the following file: nodenum-queuedOrders.dat When the resync is completed, the queued update orders are processed as normal.</td>
<td>updateLoader (on page 136)</td>
</tr>
</tbody>
</table>

**replication.def File**

**Introduction**

The *replication.def* file 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 `/IN/service_packages/SMS/etc/` directory.

### Parameters - replication.def

The `replication.def` accepts the following configurable parameters.

**COMMIT IDLE TIME**

**Syntax:**

```
COMMIT IDLE TIME=milliseconds
```

**Description:**
Timeout period (in milliseconds) for the Update Receiver (Update Loader) to become idle after an Update Request (Update Order) and commit.

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Default:** 100

**Example:**

```
COMMIT IDLE TIME=100
```

**COMMIT BUSY TIME**

**Syntax:**

```
COMMIT BUSY TIME=milliseconds
```

**Description:**
Timeout period (in milliseconds) for the Update Receiver or updateLoader to commit a change even if it remains continuously busy.

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Default:** 10000

**Example:**

```
COMMIT BUSY TIME=10000
```

**CONFIG DIR**

**Syntax:**

```
CONFIG DIR=dir
```

**Description:**
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.

**Type:** String

**Optionality:** Optional (default used if not set).

**Default:** `/IN/service_packages/SMS/etc`

**Example:**

```
CONFIG DIR=/IN/service_packages/SMS/etc
```
**CONN RETRY TIME**

Syntax: \(\text{CONN RETRY TIME}=\text{seconds}\)

Description: Time (in seconds) before an updateLoader tries to reconnect to a master replicator if none is available.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 0

Notes: If set to 0, no re-attempt is made.

Example: \(\text{CONN RETRY TIME}=0\)

---

**CONNECTION TIMEOUT**

Syntax: \(\text{CONNECTION TIMEOUT}=\text{seconds}\)

Description: Timeout (in seconds) before an attempted connection to a master is terminated and alternative is tried.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 1

Notes: Generally set to the same as HB PERIOD.

Example: \(\text{CONNECTION TIMEOUT}=1\)

---

**HB PERIOD**

Syntax: \(\text{HB PERIOD}=\text{seconds}\)

Description: Heartbeat period (in seconds)

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 10

Notes: The period should be consistent across all platforms. Not advisable to take below 3 seconds.

Example: \(\text{HB PERIOD}=10\)

---

**HB TIMEOUT**

Syntax: \(\text{HB TIMEOUT}=\text{seconds}\)

Description: Heartbeat timeout period (in seconds) used by smsMergeDaemon, before a heartbeat is considered late.

Type: Integer

Optionality: Optional (default used if not set).

Allowed:

Default: 10

Notes: Generally set to the same as HB PERIOD.

Example: \(\text{HB TIMEOUT}=10\)
HB TOLERANCE
Syntax: \texttt{HB TOLERANCE=milliseconds}
Description: Heartbeat tolerance time (in millisecs).
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 250
Notes: Default is generally used.
Example: \texttt{HB TOLERANCE=250}

HTML DIR
Syntax: \texttt{HTML DIR=dir}
Description: The directory where html files are written.
Type: String
Optionality: Optional (default used if not set).
Allowed:
Default: /IN/html
Notes: 
Example: \texttt{HTML DIR=/IN/html}

LONG TIMEOUT
Syntax: \texttt{LONG TIMEOUT=seconds}
Description: Heartbeat timeout period (in seconds) used by smsMergeDaemon to check if the connections to smsMaster and the node to be merged are stable. If they have both been responding to heartbeats within the time specified in LONG TIMEOUT, the merge takes place.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 60
Notes: 
Example: \texttt{LONG TIMEOUT=60}

MASTER PORT
Syntax: \texttt{MASTER PORT=port}
Description: The TCP port that master replicators listen for connections on.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
Default: 12343
Notes: Generally the default is used.
Example: \texttt{MASTER PORT=12343}
**MAXMASTERSNODES**

**Syntax:** `MAXMASTERSNODES=num`

**Description:** The number of master nodes used.

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Allowed:**

**Default:** 8

**Notes:**

**Example:** `MAXMASTERSNODES=8`

**MAX PENDING**

**Syntax:** `MAX PENDING=num`

**Description:** Used by master replicators to determine maximum size of their pending updates queue (the maximum number of outstanding updates that are stored before an unconnected updateLoader is considered “Out Of History”).

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Allowed:**

**Default:** 10000

**Notes:**

**Example:** `MAX PENDING=10000`

**NODE ID**

**Syntax:** `NODE ID=ID`

**Description:** Used by an updateLoader to define its replication node number. This value must be unique, and is set to the default value at installation. If more than one updateLoader is running on the same SLC machine, you must override this value with a unique number, for example, by setting the `nodeid` command line parameter.

**Type:** Integer

**Optionality:** Required

**Default:** 274

**Example:** `NODE ID=274`

**ORACLE USER**

**Syntax:** `ORACLE USER=user/pwd`

**Description:** Oracle username and associated password normally of the form user/password. Operator accounts are used to maintain security.

**Type:** String

**Optionality:** Optional (default used if not set).

**Allowed:**

**Default:** `/`

**Notes:** It is recommended that this is left as the default.

**Example:** `ORACLE USER=/`
**POLLING INTERVAL**

**Syntax:**  
POLLING INTERVAL=useconds

**Description:**  
Used to specify the polling interval (in microseconds) when the smsMaster is not receiving replication updates.

**Type:**  
Integer

**Optionality:**  
Optional (default used if not set).

**Allowed:**

**Default:**  
50000

**Notes:**

**Example:**  
POLLING INTERVAL=50000

**QUEUE WARN THRESH**

**Syntax:**  
QUEUE WARN THRESH=int

**Description:**  
The threshold intervals at which warnings are sent to the error log to indicate an increasing or decreasing pending updates queue.

**Type:**  
Integer

**Optionality:**  
Optional (default used if not set).

**Allowed:**

**Default:**  
50

**Notes:**

**Example:**  
QUEUE WARN THRESH=50

**QUEUE ERR THRESH**

**Syntax:**  
QUEUE ERR THRESH=int

**Description:**  
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.

**Type:**  
Integer

**Optionality:**  
Optional (default used if not set).

**Allowed:**

**Default:**  
200

**Notes:**  
To work correctly, this must be greater than the QUEUE WARN THRESH value.

**Example:**  
QUEUE ERR THRESH=200

**QUEUE CRIT THRESH**

**Syntax:**  
QUEUE CRIT THRESH=int

**Description:**  
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.

**Type:**  
Integer

**Optionality:**  
Optional (default used if not set).

**Allowed:**

**Default:**

**Notes:**  
To work correctly, this must be greater than the QUEUE ERR THRESH value.

**Example:**  
QUEUE CRIT THRESH=400
REP_PATH
Syntax: REP_PATH=path
Description: The directory path of the replication.config file.
Type: String
Optionality: Optional (default used if not set).
Allowed: Default: /IN/service_packages/SMS/etc/replication.config
Notes: Used by smsMergeDaemon.
Example: REP_PATH=/IN/service_packages/SMS/etc/replication.config

REPORT DIR
Syntax: REPORT DIR=dir
Description: The directory where replication reports (for example, merge reports and database comparison reports) are stored.
Type: String
Optionality: Optional (default used if not set).
Allowed: Default: /IN/service_packages/SMS/output/Replication
Notes: Example: REPORT DIR=/IN/service_packages/SMS/output/Replication

RESYNC DIR
Syntax: RESYNC DIR=dir
Description: The directory where an updateLoader's pendingUpdates.dat file is stored during a resync.
Type: String
Optionality: Optional (default used if not set).
Allowed: Default: IN/service_packages/SMS/tmp
Notes: Example: RESYNC DIR=IN/service_packages/SMS/tmp

SECONDARY DELAY
Syntax: SECONDARY DELAY=useconds
Description: Initial time (in microseconds) that the primary network has to establish a connection before attempting to connect over the secondary network as well.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 100000
Notes: A value of 0 means both networks are attempted immediately.
Example: SECONDARY DELAY=100000

SMS_PORT
Syntax: SMS_PORT=port
Description:
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 7
Default: 7
Notes: Used by smsMergeDaemon.
Example: SMS_PORT=7

STATSKEY
Syntax: STATSKEY=key
Description: Shared memory key for updateLoader replication statistics.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 270198
Default: 270198
Notes: 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.
Example: STATSKEY=270198

Example replication.def file
Here is an example replication.def file for an SMS platform:

MAX_PENDING=10000
ORACLE USER=/
HB_PERIOD=20
HB_TIMEOUT=20
LONG_TIMEOUT=60
HB_TOLERANCE=10000
CONNECTION_TIMEOUT=2
SECONDARY_DELAY=100000
CONN_RETRY=1
QUEUE_WARN_THRESH=5
POLLLING_INTERVAL=50000

CONN_RETRY_TIME=10

replication.config File

Introduction
The replication.config file is a binary configuration file that defines the current specific replication setup. It is a binary representation of the replication setup 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
- 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 Screen.

Example replication.config

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

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

--------------------------------------------------------------------
smsDumpRepConf: 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]
--------------------------------------------------------------------
smsDumpRepConf: Replication Groups configured for each node...
--------------------------------------------------------------------

NODE NUMBER [1] Prim (192.168.0.173) Sec (0.0.0.0)
 NODE NUMBER [301] Prim (192.168.0.163) 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', '') Max=('9', '')
  GROUP [ACS_CALL_PLAN_STRUCTURE] [Prim=-1] Min=('0', '') Max=('9', '')
  GROUP [ACS_CUSTOMER_CLI] [Prim=1] Min=('0', '') Max=('9', '')
  GROUP [ACS_CUSTOMER_SN] [Prim=-1] Min=('0', '') Max=('9', '')
  GROUP [ACS_LANGUAGE] [Prim=-1] Min=('0', '') Max=('9', '')
  GROUP [SMF_STATISTICS_DEFN] [Prim=-1] Min=('!', '', '') Max=('~', '~', ')
  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', '', '')
--------------------------------------------------------------------

NODE NUMBER [302] Prim (192.168.0.178) Sec (0.0.0.0)
 NODE NUMBER [301] Prim (192.168.0.163) 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', '', '') 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_CUSTOMER_SN] [Prim=-1] Min=('0', '', '') Max=('9', '', '')
  GROUP [ACS_LANGUAGE] [Prim=-1] Min=('0', '', '') Max=('9', '', '')
  GROUP [SMF_STATISTICS_DEFN] [Prim=-1] Min=('!', '', '') Max=('~', '~', ')
  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_CUSTOMER_SN] [Prim=-1] Min=('0', '', '') Max=('9', '', '')
```

38  NCC Service Management System Technical Guide
Further information

For more information, see:

- `replication.config File` (on page 37)
- `smsDumpRepConfig` (on page 159)
- `SMS User's Guide`
Overview

Introduction

This chapter explains replication check and data resynchronization processes used in SMS.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replication Checks</td>
<td>41</td>
</tr>
<tr>
<td>Database Comparisons</td>
<td>43</td>
</tr>
<tr>
<td>Database Resynchronizations</td>
<td>45</td>
</tr>
<tr>
<td>Auditing</td>
<td>47</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 diagram

Here is a diagram that shows the elements involved in replication checks.

![Replication check diagram](image)

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 SMS.</td>
<td></td>
</tr>
<tr>
<td>SCP</td>
<td>The databases on the SLCs. 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 resynchronization reports.</td>
<td></td>
</tr>
<tr>
<td>smsCompareResyncServer</td>
<td>Performs database resynchronizations and comparisons on the superior node.</td>
<td><a href="#">smsCompareResyncServer</a> (on page 149)</td>
</tr>
<tr>
<td>smsCompareResyncClient</td>
<td>Performs database resynchronizations and comparisons on the inferior node.</td>
<td><a href="#">smsCompareResyncClient</a> (on page 147)</td>
</tr>
<tr>
<td>inetCompareServer</td>
<td>Accepts Replication Check requests from the Replication Check screen.</td>
<td><a href="#">inetCompareServer</a> (on page 142)</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 run all command starts inetCompareServer as configured in the replication check report.</td>
</tr>
<tr>
<td>2</td>
<td>inetCompareServer configures and starts smsCompareResyncServer (on page 149) and writeHTMLDirFile.</td>
</tr>
<tr>
<td>3</td>
<td>smsCompareResyncClient (on page 147) handles the other end of the replication check.</td>
</tr>
<tr>
<td>4</td>
<td>writeHTMLDirFile updates index.html to include the new report for display in the screens.</td>
</tr>
</tbody>
</table>

Database Comparisons

Description

compareNode is used to initiate a full database comparison of an SCP database with the definitive copy in SMF db. This ensures that an SCP'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 SLC 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. compareServer compares all the entries of all tables which are defined to be replicated to specified updateLoader node.
Database comparison diagram

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

Database comparison components

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 SCP with the definitive copy in SMF. Starts comparisonServer.</td>
<td>compareNode (on page 140)</td>
</tr>
<tr>
<td>inputBootstrap</td>
<td>Creates configuration files for smsCompareResyncServer from replication.config.</td>
<td>inputBootstrap (on page 144)</td>
</tr>
<tr>
<td>smsMaster</td>
<td>Receives update requests and forwards them to the SMF.</td>
<td>smsMaster (on page 90)</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 resynchronizations and comparisons on the inferior node.</td>
<td>smsCompareResyncClient (on page 147)</td>
</tr>
</tbody>
</table>
### Database comparison process

The database comparison process follows these stages.

**Step** | **Action**
--- | ---
1 | `compareNode` sends a comparison request to `smsMaster`.
2 | `smsMaster` configures and starts `comparisonServer`.
3 | `inputBootstrap` provides configuration for `comparisonServer` from `replication.config`.
4 | `comparisonServer` configures and starts `smsCompareResyncServer` and `writeHTMLDirFile`.
5 | `smsCompareResyncClient` handles the other end of the comparison.
6 | `writeHTMLDirFile` updates `index.html` to include the new report for display in the screens.

### Database Resynchronizations

**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 resynchronization 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
- A resync instruction is included in a `smsTaskAgent` file

Resyncs can also be run from the command line. For more information about running resynchronizations, see `resync` (see "resyncServer" on page 146).
Database resynchronization diagram

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

Database resynchronization components

This table describes the components involved in database resynchronization 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 144)</td>
</tr>
<tr>
<td>resyncServer</td>
<td>Starts smsCompareResyncServer and inputBootstrap for resyncs.</td>
<td>resyncServer (on page 146)</td>
</tr>
<tr>
<td>smsCompareResyncClient</td>
<td>Performs database resynchronizations and comparisons on the inferior node.</td>
<td>smsCompareResyncClient (on page 147)</td>
</tr>
<tr>
<td>writeHTMLDirFile</td>
<td>Updates index.html to include new replication, comparison and resynchronization reports.</td>
<td></td>
</tr>
</tbody>
</table>
**Database resynchronization process**

The resynchronization 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 resynchronization request to smsMaster.</td>
</tr>
<tr>
<td>2</td>
<td>smsMaster will send a resynchronization 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 SCP.</td>
</tr>
<tr>
<td>8</td>
<td>smsCompareResyncServer and smsCompareResyncClient resync the databases.</td>
</tr>
<tr>
<td>9</td>
<td>smsCompareResyncServer writes a resynchronization report to /IN/html/output/SMS/resync/inferior_node_number/yyyyymmddhhmmss.report.</td>
</tr>
<tr>
<td>10</td>
<td>writeHTMLDirFile updates /IN/html/output/SMS/resync/inferior_node_number/index.html to include the new report.</td>
</tr>
</tbody>
</table>

**Auditing**

**Description**

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
- 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
- Date this change was made at

**Auditing - listAudit.sh**

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 explains the steps required to configure Service Management System.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Configuration Overview</th>
<th>49</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuring the Resource Group in the Clustered Environment</td>
<td>51</td>
</tr>
<tr>
<td>Configuring Replication Files</td>
<td>54</td>
</tr>
<tr>
<td>Configuring the Oracle Listener</td>
<td>55</td>
</tr>
<tr>
<td>Configuring the SNMP Agent</td>
<td>58</td>
</tr>
<tr>
<td>SMF AlarmMessage Format</td>
<td>64</td>
</tr>
<tr>
<td>Defining the Screen Language</td>
<td>66</td>
</tr>
<tr>
<td>Defining the Help Screen Language</td>
<td>68</td>
</tr>
<tr>
<td>Setting up the Screens</td>
<td>69</td>
</tr>
<tr>
<td>Configuring Nodes</td>
<td>73</td>
</tr>
<tr>
<td>Installing Additional Applications</td>
<td>73</td>
</tr>
</tbody>
</table>

Configuration Overview

Introduction
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  
* 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.

### Configuration components

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>
<tbody>
<tr>
<td>SMS Java Administration screens</td>
<td>SMS</td>
<td>The SMS screens provide a graphical interface for configuring many parts of SMS including:</td>
<td><strong>SMS User's Guide</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• replication</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• statistics</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• alarm filtering, and</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• reports.</td>
<td></td>
</tr>
<tr>
<td>replication.def</td>
<td>all machines with running replication agents</td>
<td>This file specifies the configuration parameters for replication. These parameters may also be specified on the command-line for each application.</td>
<td><strong>replication.def File</strong> (on page 30)</td>
</tr>
<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><strong>replication.config File</strong> (on page 37)</td>
</tr>
<tr>
<td>logjob.conf</td>
<td>all SMSs</td>
<td>This file is automatically generated when the smsSms package is installed.</td>
<td><strong>logjob.conf</strong> (on page 89)</td>
</tr>
<tr>
<td>snmp.cfg</td>
<td>all SMSs</td>
<td>This file configures the SNMP agent's details.</td>
<td><strong>Configuring the SNMP Agent</strong> (on page 58)</td>
</tr>
<tr>
<td>repsib.cfg</td>
<td>all SLCs</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 inittab 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 webserver 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/tcp"**  
Where:  
- `hostname` is the shared hostname for the SMS 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` |
Chapter 4

### Starting the smsAlarmDaemon failover

Follow these steps to start the *smsAlarmDaemon* (on page 77) failover.

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

### Starting the smsAlarmRelay failover

Follow these steps to start the *smsAlarmRelay* (on page 81) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the OracleSmsAlarmRelay directory.  
Example command:  
cd /opt/OracleSmsAlarmRelay |
| 2    | Read the readme file. |
| 3    | Change to the util directory in the OracleSmsAlarmRelay.  
Example command:  
cd util |
| 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. |

---

52  NCC Service Management System Technical Guide
## Starting the smsNamingServer failover

Follow these steps to start the `smsNamingServer` (on page 91) 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` |
| 2    | 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 94) 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` |
| 2    | 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.
- Bringing resource group `<SmsReportScheduler-harg>` online...done.

## Starting the smsReportsDaemon failover

Follow these steps to start the `smsReportsDaemon` (on page 92) 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 \textit{smsStatsThreshold} (on page 107) failover.

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

\textbf{Result:} The following information is sent to stdout:

- Creating a failover instance ...
- Registering resource type \texttt{<Oracle.SmsStatsThreshold>}...done.
- Creating failover resource group \texttt{<SmsStatsThreshold-harg>}...done.
- Creating resource \texttt{<SmsStatsThreshold-hars>} for the resource type \texttt{<Oracle.SmsStatsThreshold>}...done.
- Bringing resource group \texttt{<SmsStatsThreshold-harg>} online...done.

Starting the smsTaskAgent failover

Follow these steps to start the \textit{smsTaskAgent} (on page 109) failover.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 1    | Change to the OracleSmsTaskAgent/util directory.  
\textbf{Example command:} \texttt{cd /opt/OracleSmsTaskAgent/util} |
| 2    | Start the smsTaskAgent.  
\textbf{Example command:} \texttt{./startSmsTaskAgent} |

\textbf{Result:} The following information will be sent to stdout:

- Creating a scalable instance ...
- Registering resource type \texttt{<Oracle.SmsTaskAgent>}...done.
- Creating scalable resource group \texttt{<SmsTaskAgent-sarg>}...done.
- Creating resource \texttt{<SmsTaskAgent-sars>} for the resource type \texttt{<Oracle.SmsTaskAgent>}...done.
- Bringing resource group \texttt{<SmsTaskAgent-sarg>} online...done.

Configuring Replication Files

Introduction

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

- \texttt{replication.def}
- \texttt{replication.config}

The \texttt{replication.config} file

The \texttt{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 \textbf{Create Config File} will produce a new \texttt{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 SLCs.
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 SLC 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
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/Rep
```

**Configuring the Oracle Listener**

**Introduction**

Perform this procedure on the SMS platform only; it is not required on any of the SLCs.
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 SMS 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 either 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 ensures that the path to all the Oracle binaries is correct and that file ownership for Oracle files is preserved. |
| 2    | Go to the directory containing the listener.ora file, located in one of the following directories:  
- $ORACLE_HOME/network/admin  
- /var/opt/oracle/  
Note: This file may be located in a different place depending on the version of Oracle installed and the options selected at the time Oracle was installed. |
| 3    | Edit the listener.ora file using a text editor.  
Example:  
vi listener.ora |
| 4    | Add an ADDRESS entry to the ADDRESS_LIST.  
Where:  
- KEY=SMF  
- HOST=Hostname  
- PORT=1521  
Example:  
The example below illustrates the configuration for an SMS platform called “marco”:
LISTENER=(ADDRESS_LIST=  
 (ADDRESS=  
 (PROTOCOL=IPC)  
 (KEY=SMF)  
 )  
 (ADDRESS=  
 (PROTOCOL=tcp)  
 (HOST=marco)  
 (PORT=1521)  
 )
)  
)
Step | Action
--- | ---
5 | **Note:** 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. You should also instruct the listener to listen for requests on a designated TCP port on the SMS platform.

Add an entry to the SID_LIST

Where:
- **SID-NAME**=SMF
- **ORACLE_HOME**=oracle_version
- **GLOBAL_DBNAME**=SMF_HOSTNAME

**Example:**
The example below illustrates the configuration for an SMS platform called “marco”:

```
SID_LIST_LISTENER=(SID_LIST=
  (SID_DESC=
    (SID_NAME=SMF)
    (ORACLE_HOME=/opt/oracle/product/7.3.3)
    (GLOBAL_DBNAME=SMF.MARCO)
  )
)
```

**Note:** The listener also needs to know where it can find the information for any particular ORACLE_SID. This is accomplished through the SID_LIST. The listener needs to know the name of the SID, the Oracle home directory and the global database name.

6 | Delete the following entries:

- **USE_PLUG_AND_PLAY_LISTENER** = TRUE
- **USE_CKPFILE_LISTENER** = TRUE

**Important:** Do not edit the following settings:
- **STARTUP_WAIT_TIME_LISTENER** = 0
- **CONNECT_TIMEOUT_LISTENER** = 10

7 | Save and close the file.

8 | Type `lsnrctl stop`
The listener stops.

9 | Log on as the root user.

10 | Edit the `/IN/html/sms.jnlp` file by using a text editor.

**Example:**

```
vi /IN/html/sms.jnlp
```
Chapter 4

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 11   | **Edit the following applet parameter entries:**  
  `<param name="host" value="HOST_IP_ADDR" />`  
  `<param name="databaseID" value="LPORT:DB_SID" />`  
  **Where:**  
  - **HOST_IP_ADDR** is the host name or IP address of the SMS platform  
  - **LPORT:DB_SID** is the listener port and its SID  
  **Example:**  
  `<param name="HOST" value="marco" />`  
  `<param name="DATABASEID" value="1521:SMF" />`  
  **Note:** The installation process attempts to automatically configure this file for you, but you must check this data to ensure it is completely accurate. |
| 12   | **Save and close the file.**  
  The parameters in the **sms.jnlp** file are updated to reflect those of the listener.  
  **Note:** The **sms.html** file has been deprecated. However, if you upgraded from an earlier version of NCC, you may continue to use the **sms.html** file. You must ensure that you set parameters to the same value in both the **sms.html** file, and the **sms.jnlp** file. |

### 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*)
- Resynchronization 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
- Any triggering of the resynchronization mechanism will result in duplicate traps being forwarded to all managers.

#### Configuring the **snmp.cfg** file

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
- 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.

### SNMP relaying - switching on

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 snmp.cfg 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. |

**snmp.cfg example**

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
```

### snmp.cfg file parameters

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 ‘:’.

- **community**
  - **Syntax:** community: type
  - **Description:** The community to which smsAlarmRelay belongs.
  - **Type:** String
  - **Optionality:** Mandatory
  - **Allowed:**
  - **Default:** public
Chapter 4

Notes:

Example: community: public

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

Default:

Notes: If use-SNMPv3 (on page 62) is set to 0, the listenPort parameter has no effect.

Example: listenPort: 1161

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.

Default:

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
my-addr: 192.168.7.140

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

Notes:

Example:

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 61) 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 62) parameter is set for SNMP version 3.
- The listenPort (on page 60) parameter is configured.

Example:

opaque

Syntax: opaque: 0|1
Description: Defines encoding for SNMP specific traps.
Type: Boolean
Optionality: Mandatory if the specific (on page 62) parameter is used.
Allowed: 0 (Use if specific is set to 2 or 4).
1 (Use if specific is set to 1 or 3).
Default: Notes: The value depends on the value assigned to the specific parameter.
Example: opaque: 1

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>

Example:

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.
Example: port: 162

```
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.  

**Default:**  
**Notes:** If you use the specific parameter, you must also set the opaque (on page 61) parameter.  
**Example:** specific: 1  

```
trap
```

**Syntax:** trap: int  
**Description:** The value of the generic trap.  
**Type:** Integer  
**Optionality:** Mandatory  
**Allowed:** 6  
**Default:** 6  
**Notes:**  
**Example:** trap: 6  

```
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  
**Notes:**  
**Example:** use-SNMPv3: 1  

```
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
Allowed:
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.
Example: userName: smf_oper

Formatting as a SNMP trap message

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 SMS 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>

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.

Restarting the smsAlarmRelay

By default, SNMP trap relaying is not performed. Therefore the smsAlarmRelayStartup.sh script must be edited and the smsAlarmRelay (on page 81) 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></td>
<td>Note: The second column of the results returned is the Process Id and the third column gives the Parent Process Id.</td>
</tr>
<tr>
<td></td>
<td>Kill the Process ID from the second column.</td>
</tr>
<tr>
<td>2</td>
<td>Type kill -TERM &lt;pid&gt;</td>
</tr>
<tr>
<td></td>
<td>Result: The process is terminated and will be restarted by the init 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>(15 characters for hostname; 1 terminating character)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>time</td>
<td></td>
<td>DATE</td>
<td>not null</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>Name</td>
<td>Field size</td>
<td>Field type</td>
<td>Null value</td>
</tr>
<tr>
<td>---------------------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</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></td>
<td>DATE</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</td>
</tr>
<tr>
<td>1</td>
<td>machine</td>
</tr>
<tr>
<td>2</td>
<td>time</td>
</tr>
<tr>
<td>3</td>
<td>cpu</td>
</tr>
<tr>
<td>4</td>
<td>name</td>
</tr>
<tr>
<td>6</td>
<td>subsystem</td>
</tr>
<tr>
<td>7</td>
<td>severity</td>
</tr>
<tr>
<td>8</td>
<td>description</td>
</tr>
<tr>
<td>9</td>
<td>opcomment</td>
</tr>
<tr>
<td>10</td>
<td>count</td>
</tr>
<tr>
<td>5</td>
<td>status</td>
</tr>
<tr>
<td></td>
<td>close_time</td>
</tr>
<tr>
<td></td>
<td>change_sequence</td>
</tr>
</tbody>
</table>

**MIB field mappings - SMF_ALARM_DEFN**

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>
</tbody>
</table>
### DB Alarm

<table>
<thead>
<tr>
<th>DB Alarm</th>
<th>MIB</th>
</tr>
</thead>
<tbody>
<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 eServld) 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.

#### Language files for multi-byte character sets

To create and use a language file for a language that requires a multi-byte character set, as simplified or traditional Chinese do, as well as others, Oracle recommends that you create the file in the UTF-8 (Unicode Transformation Format-8) format.

**Note:** To support reading and writing of UTF-8 characters, you must ensure that the database character set is UTF-8. You can use the following query to determine what the database character set is:
select value from nls_database_parameters where parameter = 'NLS_CHARACTERSET';

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. You can subsequently specify a language for a specific user in the Configuration field of the User Management screen by specifying LANGUAGE=ABC where ABC must match the language file name, is case-sensitive, and does not include the file name extension. Once a language is selected for a user, it is stored in their profile.

If a character set other than UTF-8 is used to create the language file, you must specify the character set for a user using CHARSET=XYZ in the Configuration field on the User tab of the User Management screen, where XYZ specifies one of the following character sets: US-ASCII, ISO-8859-1, UTF-16BE, UTF-16LE, or UTF-16.

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 |
| 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. |
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

#### Accessing SMS

On first use of the application, the SMS user interface (UI) needs to be set up on your local machine. Use one of the following methods to open the UI from the relevant web page:

- **Java Webstart**, open the Service Management System default page on the `SMS_hostname`, then click the [WebStart link](http://SMS_hostname/sms.jnlp).
- **SMS Webstart directly:**
  
  http://SMS_hostname/sms.jnlp

Where:

- `SMS_hostname` is the hostname of an SMS in the IN.

Result: The SMS UI will start.

For more information about the SMS UI, see *SMS User's Guide*.

#### About customizing the SMS UI

You can customize the SMS UI by setting applet parameters in the `sms.jnlp` file located in the `/IN/html/` directory. You set jnlp applet parameters by using the following syntax:

```xml
<param name="parameter" value="value" />
```

Where:

- `parameter` is the name of the applet parameter
- `value` is the value to which that parameter will be set

#### About applet parameters in .html files

The ability to customize the NCC UI by setting applet parameters in the following `.html` files has been deprecated:

- `acs.html`
- `sms.html`
- `vpn.html`

If you upgraded from an earlier version of NCC, you may continue to set applet parameters in these files. However, you must ensure that any parameters that you set are also set to the same value in the corresponding `.jnlp` file:

- `acs.jnlp`
- `sms.jnlp`
- `vpn.jnlp`

Note: You use the following syntax to set applet parameters in the `.html` files:

```xml
<param name="parameter" value="value"/>
```

Where:

- `parameter` is the name of the Java applet parameter
Value is the value to which that parameter will be set

Resource properties

The following resource properties in the resources section of the sms.jnlp file define the Java 2 SE runtime environment:

java_arguments

Syntax: <property name="jnlp.packEnabled" value="true|false" />

Description: This Java resource parameter enables the use of the pack jar sig file when set to true.

Type: Boolean

Optionality: Required

Allowed: true - the Java plug-in will automatically download and try using the compressed jar file (sms.sig.jar.pack.gz).
If the browser is unable to use the compressed jar file, the uncompressed jar file (sms.jar.sig) will be downloaded.
false - will download the sms.jar.sig file.

Default: true

Notes: Use of the compressed file improves the speed of launching the application.
For more information, refer to Oracle Java SE Documentation

Example: <property name="jnlp.packEnabled" value="true" />

j2se version

Syntax: <j2se version="version" href="J2_url" />

Description: This Java resource parameter specifies the minimum Java 2 SE Runtime Environment (JRE) version that the application is supported on, and the url for Java 2 SE.

Type: String

Optionality: Required

Allowed: For more information, see Oracle Java SE Documentation.

Default: 

Notes: For more information, see Oracle Java SE Documentation.

Example: <j2se version="1.6.0+" href="http://java.sun.com/products/autodl/j2se" />

Java applet parameters

The following applet parameters are available to customize the UI:

databaseID

Syntax: <name="databaseID" value="port:sid" />

Description: The SQLNET port to use to connect to the database, and the database SID.

Type: String

Optionality: Required

Allowed: 

Default: 

Notes: 

Example: <param name="databaseID" value="1234:SMF" />
**databaseHost**

Syntax: `<name="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)

Allowed: 

Default: 

Notes: 

Example: `<param name="databaseHost" value = "192.0.2.0:1234:SMF" />`

**clusterDatabaseHost**

Syntax: See example

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

Allowed: 

Default: 

Notes: 

Example: `<param name="clusterDatabaseHost" value = "(DESCRIPTION=(LOAD_BALANCE=YES)(FAILOVER=ON)(ENABLE=BROKEN)(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)))" />`

**logo**

Syntax: See example

Description: Sets the system graphic that displays briefly in a splash window immediately before the Login window displays. At installation, the logo parameter is set to the .gif file for the Oracle logo.

Type: String

Optionality: Optional

Allowed: A valid network path/file.

Default: None

Notes: Set to the relative path name for the logo .gif file to use.

Example: `<param name="logo" value="SMS/images/oracle.gif" />`

**MAX_CONTROL_PLANS_DISPLAYED**

Syntax: See example

Description: Sets the maximum number of control plans visible in the search box.

Type: String

Optionality: Optional
Chapter 4

Allowed:

Default:

Notes:

Example:  
<param name="MAX_CONTROL_PLANS_DISPLAYED" value="200" />

TZ

Syntax:  See example

Description:  The screens in the NCC UI display all time and date values in the time zone set by this parameter.

Type:  String

Optionality:  Optional (default used if not set)

Allowed:  Any Java supported time zone.

Default:  GMT

Notes:  For a full list a Java supported time zones see Time Zones in the appendix in ACS Technical Guide.

Example:  
<param name="TZ" value="GMT" />

Java Webstart

To launch GUI applications using Java Webstart, you must ensure the Web server supports the jnlp file type.

For example, if the Web server is Apache, find the mime.types file (default location is /etc/apache/mime.types), add the following line in the file, then restart Apache.

application/x-java-jnlp-file jnlp

Example jnlp resources and applet parameters

Here is an example of the parameters for resources and applet in the sms.jnlp file. Note that other applications, such as ACS and CCS may add parameters to this file.

<jnlp spec="1.0+"
    codebase="http://HOST_IP_ADDR/"
    href="sms.jnlp" >
    ...
    ...
<resources>
    <j2se version="1.6.0+" href="http://java.sun.com/products/autodl/j2se" />
    <property name="jnlp.packEnabled" value="true" />
    <jar href="sms.sig.jar" />
    <jar href="common.sig.jar" />
    <jar href="ojdbc6.jar.sig" />
    <extension name="Oracle Help for Java" href="ohj.jnlp" />
    ...
    ...
    <property name="java.util.Arrays.useLegacyMergeSort" value="true" />
</resources>

<applet-desc
    documentBase="http://HOST_IP_ADDR"
    name="SMSApplet"
    main-class="UserScreens.Applet1"
    width="275"
    height="25" >
    <param name="TZ" value="GMT" />
    <param name="host" value="HOST_IP_ADDR" />

Configuring Nodes

SMS Nodes

During installation of the SMS software, each SMS 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*.

SLC Nodes

In a clustered installation, each SLC 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

Installing the 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 SLCs 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 SLCs (the i+256 node).</td>
</tr>
</tbody>
</table>
Order of replication

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

Introduction

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

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.

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

In this chapter

This chapter contains the following topics.

- cmnConfigRead 76
- cmnReceiveFiles 76
- smsAlarmDaemon 77
- smsAlarmManager 79
- smsAlarmRelay 81
- smsConfigDaemon 83
- smsConfigDaemonScript 84
- smsCdrProcess.sh 86
- smsDbCleanup.sh 87
- smsLogCleaner 88
- smsMergeDaemon 89
- smsMaster 90
- smsNamingServer 91
- smsReportsDaemon 92
- smsReportScheduler 94
- smsReportCleanupStartup.sh 96
- smsStatsDaemon 97
- smsStatisticsWriter 97
- smsStatsThreshold 107
- smsSendConfig.sh 109
- smsTaskAgent 109
- smsTrigDaemon 112
cmnConfigRead

Purpose

CmnConfigRead is used by the installation process to read the configuration files.
CmnConfigRead reads the NCC 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:
FILENAME=`cmnConfigRead CCS.MyReport.filename /IN/service_packages/CCS/tmp/MyReport.log`
This 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.

CmnReceiveFiles

Purpose

CmnReceiveFiles collects EDRs from cmnPushFiles and writes them to the specified directory on the SMS.

Startup

CmnReceiveFiles is started by the following entry in /etc/inetd.conf:

smsoperFile stream tcp nowait smf_oper /IN/service_packages/SMS/bin/cmnaReceiveFilesStartup.sh

Parameters

CmnReceiveFiles does not have any direct parameters or configuration. Most details are provided by cmnPushFiles with the EDR.
The port cmnReceiveFiles listens on is set in /etc/services in the following line:

smsoperFile 2028/tcp # cmnAddInetServicesEntry

Important: The port number must match the port specified by cmnPushFiles.

Failure

If cmnReceiveFiles fails, the EDRs will stay on the SLC and be moved to the retry directory. For more information about this process, see cmnPushFiles (on page 118).

Output

CmnReceiveFiles writes the EDRs to the directory specified by cmnPushFiles.
smsAlarmDaemon

Purpose
The smsAlarmDaemon runs on all alarm-managed nodes in the SMS system, including the SMS 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 SMS 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 inititab, through the /IN/service_packages/SMS/bin/smsAlarmDaemonSmsStartup.sh shell script.

In a clustered install, this task is started by the clustering software, through the /IN/service_packages/SMS/bin/smsAlarmDaemonCluster.sh shell script.

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 path</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>
## Parameter Default Description

<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 seconds</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). <strong>Note:</strong> Only valid when used in conjunction with -f.</td>
</tr>
<tr>
<td>-l seconds</td>
<td>2</td>
<td>Filter Period - duration between linked-list checks (seconds).</td>
</tr>
<tr>
<td>-n number</td>
<td>5</td>
<td>Filter Number - number of alarm messages allowed within the time period. <strong>Allowed values:</strong> integers</td>
</tr>
<tr>
<td>-m number</td>
<td>1000</td>
<td>Maximum number of alarm messages to buffer. <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 SMS 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 node</td>
<td>Direct to the Oracle DB</td>
<td>Rep Node - specify replication requester node.</td>
</tr>
<tr>
<td>-t seconds</td>
<td>1</td>
<td>Commit Interval - maximum interval between committing to the database (seconds).</td>
</tr>
<tr>
<td>-u user/pass</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/smsAlarmDaemonSms.log.

smsAlarmManager

Purpose

The smsAlarmManager runs on the SMS. 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, through the /IN/service_packages/EFM/bin/smsAlarmManagerStartup.sh shell script.

The inittab entry will be similar to that shown below:

```bash
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 2>&1 0<&1
```

Configuration

The smsAlarmManager accepts the following command line arguments.

Usage:

```bash
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 user/password
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a alarm_batch_size</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 correlate_batch_size</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 timeout_commit_rate</td>
<td>1000</td>
<td>The number of rows to update with automatic timeout before committing.</td>
</tr>
</tbody>
</table>
### Parameter Default Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-p pending_timeout_length</code></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><code>-r reload_defn_interval</code></td>
<td>86400</td>
<td>Interval (s) for reloading the alarm definitions for the DB. Interval between reloading the regular expressions from SMF_ALARM_DEFN and SMF_ALARM_IGNORE. This should only be needed after an install of new packages/patches, and also acts to keep the preferred cache current</td>
</tr>
<tr>
<td><code>-s</code></td>
<td>50000</td>
<td>Interval (microseconds) to sleep for when no work to do</td>
</tr>
<tr>
<td><code>-t timeout_check_interval</code></td>
<td>300</td>
<td>Interval between checks for alarms that need to be closed with a timeout.</td>
</tr>
<tr>
<td><code>-u &quot;user/password&quot;</code></td>
<td>/</td>
<td>The username/password combination used to log into the database. 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:

```bash
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
```
smsAlarmRelay

Purpose

The smsAlarmRelay is responsible for implementing the SNMP Agent (on page 58). 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

Startup

In an unclustered installation, this task is started by entry sms1 in the initdb, through the /IN/service_packages/SMS/bin/smsAlarmRelayStartup.sh shell script.

In a clustered installation, this task is started by the cluster software, through the /IN/service_packages/SMS/bin/smsAlarmRelayCluster.sh shell script.

Parameters

The smsAlarmRelay accepts the following command line arguments.

Usage:


Note: SNMP processing is not currently enabled by default.

The available parameters are:

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

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

Allowed:

Default: 1

Notes:

Example:

-\texttt{p}

Syntax: \texttt{-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

Notes:

Example:

-\texttt{x}

Syntax: \texttt{-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

Notes:

Example:

-\texttt{t}

Syntax: \texttt{-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

Notes:

Example:

-\texttt{e}

Syntax: \texttt{-e}

Description: Loads the EFM rules from the smf_alarm_relay_filter database table to provide alarm filtering.

Type: Boolean

Optionality: Optional.
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 off line 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 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.

Output

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 table summarizes 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>Host name of the target SNMP TRAPS recipient</td>
</tr>
<tr>
<td>FILE</td>
<td>Name of a file to which the daemon has write access</td>
</tr>
<tr>
<td>NFM</td>
<td>Host name of the NFM target.</td>
</tr>
<tr>
<td>Q3</td>
<td>Host name of the Q3 target.</td>
</tr>
<tr>
<td>SNMP</td>
<td>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, SMS) as well as the target node (example, SLC). 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 the \(-\text{m}\) parameter is missing, smsConfigDaemon monitors for changes to the derived \texttt{eserv.config}\ file (example, \texttt{eserv.config.derived}) on the target node, and calls \texttt{smsApplyConfig.sh} if it finds changes to the file.

**Startup**

smsConfigDaemon is started by the script \texttt{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 \texttt{smsConfigDaemonScript Configuration} (on page 84).

**Failure**

If the smsConfigDaemon fails, the secondary scripts, \texttt{smsSendConfig.sh} and \texttt{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 /\texttt{IN/service_packages/SMS/tmp/smsConfigDaemonMaster.log} if they reside on the target node to /\texttt{IN/service_packages/SMS/tmp/smsConfigDaemonClient.log}.

### \texttt{smsConfigDaemonScript}

**Purpose**

\texttt{smsConfigDaemonScript} is responsible for starting the smsConfigDaemon process, depending on the supplied parameter (if applicable). It also loads the \texttt{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 \texttt{smsConfigDaemon} (on page 83).

**Startup**

\texttt{smsConfigDaemonScript} is started by the system and is not intended to be changed by the user.

**Configuration**

\texttt{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>\texttt{smp_db_user}</td>
<td>user/password</td>
<td>Oracle user/password for the SMF.</td>
</tr>
<tr>
<td>\texttt{user/password}</td>
<td></td>
<td>Example: \texttt{smf/smf}</td>
</tr>
<tr>
<td>\texttt{scp_db_user}</td>
<td>user/password</td>
<td>Oracle user/password for the smsSignalConfigChange script should use for sqlplus.</td>
</tr>
<tr>
<td>\texttt{user/password}</td>
<td></td>
<td>Example: \texttt{scp/scp}</td>
</tr>
<tr>
<td>\texttt{connect_string}</td>
<td></td>
<td>Any extra connect parameters to be used by smsConfigDaemon</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>---------</td>
<td>-------------</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>esgConfigMaster</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>
<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_full_path</td>
<td></td>
<td>Full path of the derived config file monitored by the SCP config daemon (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 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, when smsConfigDaemon has been started using the --m option, to the /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

Purpose

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 smsProcessCdr (on page 162).

EDR format

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 "Pipe Tag LF" format, which uses \texttt{TAG=VALUE} pairs separated by the "|" character. Records are line field delimited.

For more information about this format, see the ACS Technical Guide.

Startup

This task is run in the crontab for smf_oper, by default at 1:00 am system clock time. It is scheduled as the /IN/service_packages/SMS/bin/smsCdrProcess.sh script.

The script runs the smsProcessCdr process with set parameters.

Configuration

The following command in the smsProcessCdr.sh prevents the EDR from being processed and copies it directly to the output directory.

\texttt{Example Command: $BINDIR/smsProcessCdr -d $CDRDIR -D $OUTDIR -s $INSFX -S $OUTSFX}

To process EDRs, use the following command instead:

\texttt{Example Command: $BINDIR/smsProcessCdr -t $OUTFMT -d $CDRDIR -D $OUTDIR -s $INSFX -S $OUTSFX}

Failure

If the process is not running, EDR files will build up in the /IN/service_packages/SMS/cdr/received directory. The filesystem usage will rise above standard operational levels.

Output

The smsCdrProcess.sh writes error messages to the system messages file, and also writes additional output to /IN/service_packages/SMS/tmp/smsCdrProcess.sh.log.
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. Refers to the actual age of the alarm and controls the clean up all alarms of a certain age, regardless of whether they are noted.</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>alarmNotedAge</td>
<td>3</td>
<td>Controls the clean up of noted (closed) alarms.</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 <code>smf_alarm_unknown</code>. 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.
Chapter 5

**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><code>-c &lt;configuration file name&gt;</code></td>
<td><code>logjob.conf</code></td>
<td>The configuration file to use.</td>
</tr>
<tr>
<td><code>-d &lt;days&gt;</code></td>
<td><code>7</code></td>
<td>How often to clean archive in days.</td>
</tr>
<tr>
<td><code>-s &lt;storage file name&gt;</code></td>
<td><code>storage.txt</code></td>
<td>The storage file to use.</td>
</tr>
<tr>
<td><code>-h</code></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`. 
logjob.conf

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>log &lt;file&gt;</td>
<td>The full directory path and name of the file to be cleaned. You can include the '*' wildcard in the file name if required.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong> log /IN/service_packages/SMS/tmp/smsNamingServer.log</td>
</tr>
<tr>
<td></td>
<td><strong>Tip:</strong> Most processes and tools document where their output is written to in their Output topic.</td>
</tr>
<tr>
<td>age &lt;hrs&gt;</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></td>
<td><strong>Example:</strong> age 100</td>
</tr>
<tr>
<td>size &lt;size&gt;</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></td>
<td><strong>Examples:</strong> size 60K, or size 60M</td>
</tr>
<tr>
<td>arcdir &lt;dir&gt;</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></td>
<td><strong>Example:</strong> arcdir /IN/service_packages/SMS/tmp/archive</td>
</tr>
<tr>
<td>logonce</td>
<td>Only specify this parameter if you just want to keep one archived version of the log file.</td>
</tr>
</tbody>
</table>

smsMergeDaemon

**Purpose**

The smsMergeDaemon monitors the connections between the SMS and SLCs via a heartbeat. The smsMergeDaemon will initiate a startMerge to resynchronise the SMS and SLCs where:

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

For more information about the startMerge process, see `startMerge` (on page 171).

**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 30). 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 resynchronization 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, through 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

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

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, through the shell script:

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

In a clustered installation this task is started by the cluster software, through 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
Syntax:  -u usr/pwd
Description:  The Oracle user and password pair.
Type:  String
Optionality: Optional (default used if not set).
Allowed: /
Default: /
Notes:
Example: -u /

-p
Syntax: -p port
Description: The port number on which to listen for requests.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 
Default: 7362
Notes:
Example: -p 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 SMS 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 inittab, 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>

92  NCC Service Management System Technical Guide
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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed value type: ASCII String</td>
</tr>
<tr>
<td>-p &lt;port&gt;</td>
<td>7362</td>
<td>smsNamingServer port number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed value type: ASCII String</td>
</tr>
<tr>
<td>-u &lt;user/password&gt;</td>
<td></td>
<td>Oracle SMF database username and password.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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>
<tr>
<td>-m</td>
<td>2</td>
<td>Maximum number of concurrent reports per node.</td>
</tr>
</tbody>
</table>

**Failure**

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.

**Output**

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 file name for the report output is in the format:

YYMMDDHHmmss.9_random_characters.txt

**Interactive reports**

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 SMS host, and hence the CORBA service port must be fixed. Use the "s" parameter for this purpose.

**smsReportScheduler**

**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*.

**Startup**

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
```

**Parameters**

The smsReportScheduler supports the following command-line options:

**Usage:**

```
```

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).

**Allowed:**

**Default:** `/IN/service_packages/SMS/input`

**Example:**

**-o <dir>**

**Syntax:** `-o <dir>`

**Description:** The output directory for report generation.

**Type:** String

**Optionality:** Optional (default used if not set).
Chapter 5

Background Processes on the SMS

Allowed:
Default: /IN/service_packages/SMS/output
Notes: Report may provide a default output directory which overrides smsReportDaemon's default.

Example:

-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).
Allowed: Default:
Notes:
Example:

-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
Notes:
Example:

-z <timezone>

Syntax: -z <timezone>
Description: The timezone in which to schedule the report.
Type: String
Optionality: Optional (default used if not set).
Allowed: Java supported timezone
Default: GMT
Notes: For a full list of Java supported timezones see ACS Technical Guide - Appendix TimeZones.
Example: -z "EST"

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>
<tr>
<td>sendmail (or equivalent delivery agent)</td>
<td>E-mail delivery agent.</td>
<td>daemon started at boot time.</td>
</tr>
<tr>
<td>lpr</td>
<td>Printing utility.</td>
<td>/usr/ucb/lpr</td>
</tr>
</tbody>
</table>

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

Purpose

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.

Startup

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:

/IN/service_packages/SMS/bin/smsReportsCleanerStartup.sh

Parameters

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.

Failure

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

/IN/service_packages/SMS/output

Output

The smsReportsCleaner run by smf_oper writes error messages to the system messages file, and also writes additional output to /IN/service_packages/SMS/tmp/smsReportsCleanerStartup.sh.log.
smsStatsDaemon

Description

smsStatsDaemon can be run as a background process on an SMS, SLC and also on other IPs such as Voucher and Wallet Servers.

For more configuration information about the process and its parameters, see smsStatsDaemon (on page 128).

smsStatisticsWriter

Purpose

The smsStatisticsWriter is responsible for collecting statistical data provided by the smsStatDaemons and writing it to files, either for standalone statistics, or for groups of statistics associated with an event.

smsStatisticsWriter structure

Here is an example structure of the smsStatisticsWriter.config file.

```plaintext
smsStatisticsWriter = {
    tempDir = "/IN/service_packages/SMS/tmp/smsStatisticsWriter"
    outDir = "/IN/service_packages/SMS/logs/smsStatisticsWriter"
    outDirType = 'FLAT'
    outDirExpectedFiles = 65536
    outDirBucketSize = 10
    outFileName = "smsStatisticsWriter"
    maxFileSize = 100
    maxFileOpenTime = 3600
    statsDaemonRestartDelay = 10
    statsDaemonStartupTime = 5
    scanInterval = 100000
    replicationAllowance = 60
    endOfEventTolerance = 5
    resetInterval = {
        days = 1
        hours = 0
        minutes = 0
        seconds = 0
    }
    Events = [
        {
            eventName = "EventGood"
            resetAllEventStatisticsOnStartup = true
            eventResetBaseTime = "20110511130000"
            resetInterval = {
                days = 0
                hours = 0
                minutes = 4
                seconds = 0
            }
            eventStartDateTime = "20100728000000"
            eventEndDateTime = "20101231000000"
            eventWritePeriod = 30
            Statistics = [
                {
                    applicationName = "TELEVOTING"
                }
            ]
        }
    ]
}
```
smsStatisticsWriter parameters

The smsStatisticsWriter section accepts the following parameters.

tempDir

Syntax:  

tempDir = "dir"

Description:  The temporary directory for writing intermediate statistics files to.

Type:  String

Optionality:  Optional (default used if not set).

Allowed:  

Default:  

"/IN/service_packages/SMS/tmp/smsStatisticsWriter"

Notes:  

Example:  

tempDir = "/IN/service_packages/SMS/tmp/smsStatisticsWriter"

outDir

Syntax:  

outDir = "dir"

Description:  The base directory for storing completed statistics files.
Chapter 5

Background Processes on the SMS

Type: String
Optionality: Optional (default used if not set).
Default: "/IN/service_packages/SMS/logs/smsStatisticsWriter"
Notes: Example: outDir = "/IN/service_packages/SMS/logs/smsStatisticsWriter"

outDirType
Syntax: outDirType: = "type"
Description: File store type.
Type: String
Optionality: Optional (default used if not set).
Allowed: HASH or FLAT
Default: "FLAT"
Notes: Example: outDirType: = "FLAT"

outDirExpectedFiles
Syntax: outDirExpectedFiles = num
Description: The maximum number of files in outDir when outDirType is HASH.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 65536
Notes: Example: outDirExpectedFiles = 65536

outDirBucketSize
Syntax: outDirBucketSize = size
Description: The maximum number of files in any leaf directory when outDirType is HASH.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 10
Notes: Example: outDirBucketSize = 10

outFileName
Syntax: outFileName = "name"
Description: The base of the statistics file name.
Type: String
Optionality: Optional (default used if not set).
Chapter 5

Default: "smsStatisticsWriter"
Notes:
Example: outFileName = "smsStatisticsWriter"

maxFileSize
Syntax: maxFileSize = size
Description: The maximum size of a statistics file (kilobytes).
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 100
Notes:
Example: maxFileSize = 100

maxFileOpenTime
Syntax: maxFileOpenTime = sec
Description: The maximum time a statistics file can be kept open (seconds).
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 3600
Notes:
Example: maxFileOpenTime = 3600

statsDaemonRestartDelay
Syntax: statsDaemonRestartDelay = sec
Description: The length of time to allow updated statistics to be replicated to the SLC nodes before restarting the smsStatsDaemon processes on those nodes. (seconds).
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 10
Notes:
Example: statsDaemonRestartDelay = 10

statsDaemonStartupTime
Syntax: statsDaemonStartupTime = <seconds>
Description: The length of time (in seconds) to allow for smsStatsDaemon to restart and initialize after startUp request issued.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 5
Notes:
Example: statsDaemonStartupTime = 8
scanInterval
Syntax: \( \text{scanInterval} = \text{msec} \)
Description: Length of time between scans of the statistics (microseconds).
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 100000
Notes: Example: \( \text{scanInterval} = 100000 \)

replicationAllowance
Syntax: \( \text{replicationAllowance} = <\text{seconds}> \)
Description: The time (in seconds) to allow for stats to be replicated to SMS from SLC.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 60
Notes: For best results this value should be less than any of the reporting periods of statistics included in event configuration.
Example: \( \text{replicationAllowance} = 40 \)

Events
Event configuration parameters may or may not have default values, where defaults are defined that parameter may be omitted, for all others the parameter is mandatory. There may be multiple events configured.

Each event accepts the following parameters.

Events = [
    {
        eventName = "EventGood"
        resetAllEventStatisticsOnStartup = true
        eventResetBaseTime = "20110511130000"
        resetInterval = {
            days = 0
            hours = 0
            minutes = 4
            seconds = 0
        }
        eventStartDateTime = "20100728000000"
        eventEndDateTime = "20101231000000"
        eventWritePeriod = 30
        Statistics = [
            {
                applicationName = "TELEVOTING"
                statisticName = "Stat1"
            }
        ]
    }
]

eventName
Syntax: eventName = "name"
Description: The name of the event.
Type: String
resetAllEventStatisticsOnStartup
- Syntax: `resetAllEventStatisticsOnStartup = true|false`
- Description: Reset all statistics defined for the event on start up.
- Type: Boolean
- Optionality: Optional (default used when omitted)
- Allowed: true, or false
- Default: true
- Notes: If true, individual statistic resetStatisticOnStatup config is overridden.
- Example: `resetAllEventStatisticsOnStartup = false`

eventResetBaseTime
- Syntax: `eventResetBaseTime = "<date and time>"`
- Description: Sets a specific date and time base to calculate resets for the statistics configured for this event.
- Type: String
- Optionality: Optional (default used if not set).
- Allowed: 
- Default: "20100101000000" 00:00:00 on Jan 1 2010
- Notes: Date and time format is yyyymmddHHMMSS
- Example: `eventResetBaseTime = "20110511130000"

resetInterval
- Syntax: `resetInterval = {<days>,<hours>,<minutes>,<seconds>}`
- Description: The interval between periodic resets of all statistics configured for the event.
- Type: Parameter list
- Optionality: Optional (default used if missing)
- Allowed: Global resetInterval values are used.
- Default: If non zero, the reset period configured for individual statistics will be overridden.
- Notes: 
- Example: `resetInterval = {
    days = 0
    hours = 0
    minutes = 4
    seconds = 0
  }

eventStartDateTime
- Syntax: `eventStartDateTime = "dateandtime"
- Description: The event start time.
Event statistics

Each event can collect more than one statistic.

For example the event "EventGood" is a contest where you can vote for three contestants. The statistic for each contestant is collected separately to determine the winner.

Each event statistic accepts the following parameters:

applicationName

Syntax: applicationName = "name"
Description: The name of application to which the statistic belongs.
Type: String
Optionality: Mandatory
Allowed: Expected format yyyymmddHHMMSS
Default: no default
Notes:
Example: applicationName = "TELEVOTING"

statisticName
Syntax: statisticName = "name"
Description: The name of the statistic.
Type: String
Optionality: Mandatory
Allowed: Default: no default
Notes: Example: statisticName = "Stat1"

Statistics
As well as event statistics, smsStatisticsWriter statistics as a whole are collected, irrespective of events. Each statistic accepts the following parameters:

applicationName
Syntax: applicationName = "name"
Description: The name of application to which the statistic belongs.
Type: String
Optionality: Mandatory
Allowed: Default: no default
Notes: Example: applicationName = "TELEVOTING"

statisticName
Syntax: statisticName = "name"
Description: The name of the statistic.
Type: String
Optionality: Mandatory
Allowed: Default: no default
Notes: Example: statisticName = "Stat1"

resetStatisticOnStartup
Syntax: resetStatisticOnStartup = true|false
Description: Reset statistic on start up.
Type: Boolean
Optionality: Optional (default used if omitted)
Allowed: Default: true
Notes:
Example: resetStatisticOnStartup = false

statWritePeriod

Syntax: statWritePeriod = <seconds>
Description: The period (seconds) between successive writes of statistics data to the report file.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default:
Default: 0
Notes: When set to 0 the write period is synchronized to the sample period defined for the statistic.
Example: statWritePeriod = 0

statResetBaseTime

Syntax: statResetBaseTime = "<date and time>"
Description: Sets a specific date and time for setting the statistic to zero.
Type: String
Optionality: Mandatory
Allowed: Default: None
Notes: Date and time format is yyyyMMddHHMMSS
The statistic resetInterval parameter timing starts from this parameter's date and time.
Example: statResetBaseTime = "20110511130000"

resetInterval

Syntax: resetInterval = {<days>,<hours>,<minutes>,<seconds>}
Description: The interval between periodic resets of the statistic.
Type: Array of parameters
Optionality: optional (default used if not set).
Allowed: Default: Global resetInterval values are used.
Notes: This interval is started from the date and time statResetBaseTime if it is present.
Example: resetInterval = {
        days = 0
        hours = 0
        minutes = 4
        seconds = 0
    }

Event name file format

The smsStatisticsWriter process creates a file format as detailed below for an event configured in SMS.smsStatisticsWriter.Events.eventName:

"Event Name"
"Start Time"
"Stop Time"

[Automatic statistic reset Time Stamp]

Timestamp

[Automatic statistic reset Time Stamp]

"Statistic ID" Statistics Count

[Automatic statistic reset Time Stamp]

"Statistic ID" Statistics Count

....

Timestamp

[Automatic statistic reset Time Stamp]

"Statistic ID" Statistics Count

[Automatic statistic reset Time Stamp]

"Statistic ID" Statistics Count

For example:

[Automatic statistic reset Time Stamp]

Timestamp

"Statistic ID" Statistics Count

[Automatic statistic reset Time Stamp]

"Statistic ID" Statistics Count

<table>
<thead>
<tr>
<th>Timestamp</th>
<th>&quot;Contestant A&quot;</th>
<th>&quot;Contestant B&quot;</th>
<th>&quot;Contestant C&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>21-04-2010 20:00:00</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21-04-2010 20:01:00</td>
<td>300</td>
<td>20</td>
<td>50</td>
</tr>
<tr>
<td>21-04-2010 20:20:00</td>
<td>1200</td>
<td></td>
<td>120</td>
</tr>
<tr>
<td>21-04-2010 20:20:00</td>
<td></td>
<td>0</td>
<td>95</td>
</tr>
<tr>
<td>21-04-2010 20:30:00</td>
<td>15345</td>
<td>12789</td>
<td>120</td>
</tr>
</tbody>
</table>
Statistics file format

The smsStatisticsWriter process creates a file format as detailed below for a statistic configured in SMS.smsStatisticsWriter.Statistics:

```
=======================================
"Application ID"
"Statistic ID"
=======================================
[Automatic statistic reset Timestamp]
Timestamp Statistics Count
...
[Automatic statistic reset Timestamp]
Timestamp Statistics Count
...
[Automatic statistic reset Timestamp]
Timestamp Statistics Count
```

For example:

```
=======================================
"TPSA"
"Service A"
=======================================
01-01-2010 00:00:00  0
01-01-2010 00:00:30  2
...
Automatic statistic reset 31-01-2010 12:00:00
31-01-2010 12:00:00  0
31-01-2010 12:00:30  3
...
31-01-2010 23:00:00  4000
```

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

Purpose

smsSendConfig.sh resides on the source node (for example, an SMS) and performs the following functions:

- archives current master XML file
- stores audit information
- sets link to current archived file
- converts XML format to derived eserv.config using the script cmnConfigXmlConvert.sh, and
- sends derived eserv.config file to target node using scp.

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 84).

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.

Output

The smsSendConfig.sh 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.

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 SMS host, and hence the CORBA service port must be fixed. Use the "-s" parameter for this purpose.
Chapter 5

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:


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).
Allowed: 
Default: Use standard connection and copy.
Notes:
Example:

-i <host>

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

-p <port>

Syntax: -p <port>
Description: The port on the IOR Listener host to connect to.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 
Default: 5556
Notes:
Example: -p 13579
-u <usr/pwd>
Syntax: -u <usr/pwd>
Description: The userid and password to use to log into Oracle.
Type: String
Optionality: Optional (default used if not set).
Allowed: 
Default: / 
Notes: 
Example:

-t <host>
Syntax: -t <host>
Description: The ILU transport host to connect to.
Type: String
Optionality: Optional (default used if not set).
Allowed: 
Default: NULL 
Notes: 
Example:

-s <port>
Syntax: -s <port>
Description: The port on the ILU transport host to connect to.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 
Default: 0 
Notes: 
Example:

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

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.
Chapter 5

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 SMS 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 SLC 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/ESERV/SmstrigDaemon/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 SMSs.

Parameters

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

Usage:

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

oracleLogin

**Syntax:** \( \text{oracleLogin} = "<\text{usr}>/<\text{pwd}>" \)

**Description:** The Oracle user name and password that smsTrigDaemon uses when connecting to the database.

**Type:** String

**Optionality:** Mandatory

**Allowed:**

**Default:** /

**Notes:**

**Example:**

useORB

**Syntax:** \( \text{useORB} = <\text{true}|\text{false}> \)

**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

**Notes:**

**Example:**

listenPort

**Syntax:** \( \text{listenPort} = <\text{port}> \)

**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.

**Example:**

slcBusyTimeout

**Syntax:** \( \text{slcBusyTimeout} = \text{seconds} \)

**Description:** The number of seconds before connections to the SLC or VWS nodes will time out.

**Type:** Integer

**Optionality:** Optional (default used if not set).

**Allowed:**

**Default:** 10

**Notes:**

**Example:**

slcBusyTimeout = 15
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 114) parameter array is ignored.
- If set to false and useORB (on page 113) is also set to false, smsTrigDaemon will do nothing.
Example:

extraFIFO
Syntax: extraFIFO = [ 
  "<dir>"
  ...
]
Description: The paths that smsTrigDaemon should create for extra FIFOs.
Type: Parameter array
Optionality: Optional
Allowed:
Default: Empty set
Notes:
- If useFIFO (on page 114) is set to false, this parameter array is ignored.
Example:

scps
Syntax: scps = [ 
  "<ip>;<port>"
  ...
]
Description: The IP address and port number for each SLC 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:

```python
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
```

Control plan execution requests

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

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>smsTrigDaemon attempts to connect to one of the SLCs in the following way.</td>
</tr>
<tr>
<td></td>
<td>- If a connection to an SLC is established and not in use, that connection is used.</td>
</tr>
<tr>
<td></td>
<td>- If a connection to an SLC is not established, smsTrigDaemon attempts to open one.</td>
</tr>
<tr>
<td></td>
<td>- SLCs are identified by the <code>scps</code> configuration parameter.</td>
</tr>
<tr>
<td></td>
<td>- smsTrigDaemon polls this list until it finds an available connection.</td>
</tr>
<tr>
<td></td>
<td>- If the connection fails, the next SLC in the list is tried.</td>
</tr>
<tr>
<td>2</td>
<td>smsTrigDaemon connects to the SLC using either port 3072 or another port specified by the <code>scps</code> parameter.</td>
</tr>
<tr>
<td></td>
<td>- If a connection to an SLC is established and not in use, that connection is used.</td>
</tr>
<tr>
<td></td>
<td>- SLCs are identified by the <code>scps</code> configuration parameter.</td>
</tr>
<tr>
<td></td>
<td>- smsTrigDaemon polls this list until it finds an available connection.</td>
</tr>
<tr>
<td></td>
<td>- If the connection fails, the next SLC in the list is tried.</td>
</tr>
<tr>
<td>3</td>
<td>smsTrigDaemon sends an XML message via an HTTP &quot;POST / HTTP/1.1&quot; request.</td>
</tr>
<tr>
<td></td>
<td>- The syntax of the message is:</td>
</tr>
<tr>
<td></td>
<td><code>&lt;control-plan-exec&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;control-plan&gt;&lt;name of control plan&gt;&lt;/control-plan&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;service-handle&gt;&lt;service handle&gt;&lt;/service-handle&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;cgpn&gt;&lt;calling party&gt;&lt;/cgpn&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;cdpn&gt;&lt;called party&gt;&lt;/cdpn&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;ext id=&quot;400&quot;&gt;&lt;host&gt;&lt;/ext&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;ext id=&quot;401&quot;&gt;&lt;user&gt;&lt;/ext&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>&lt;ext id=&quot;402&quot;&gt;&lt;more extensions&gt;&lt;/ext&gt;</code></td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>&lt;/control-plan-exec&gt;`</td>
</tr>
</tbody>
</table>

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 SLCs of an smsTrigDaemon request.

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

Because it is not possible to know in advance which SLC will be selected by smsTrigDaemon, a data consistency check is performed on all replicated SLCs. 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>
</tbody>
</table>
| 3     | The Java management screen extracts relevant information from the report, including:  
|       | • The number of nodes checked.  
|       | • The number of nodes that failed the check.  
|       | • The number of nodes that replied to the check request.  
|       | • The number of nodes that reported inconsistent data. |
| 4     | The consistency check:  
|       | • succeeds, if:  
|       |   ▪ at least one node replied without error, and  
|       |   ▪ no node reported inconsistent data.  
|       | • fails, if:  
|       |   ▪ no nodes replied without error, or  
|       |   ▪ one or more nodes reported inconsistent data. |
| 5     | If the check succeeds, the request proceeds. |
| 6     | 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. |
Background Processes on the SLC

Overview

Introduction

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

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 prerequisite 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.

- smsApplyConfig.sh
- cmnPushFiles
- infMaster
- smsAlarmDaemon
- smsLogCleaner
- smsStatsDaemon
- updateLoader

smsApplyConfig.sh

Purpose

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

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.
- Signals changes using smsSignalConfigChanges.sh.

If a SIGHUP is required, smsSignalConfigChanges.sh will in turn call smsSendSighup.sh (which will run with root permissions).
Chapter 6

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 84).

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 SMS from SLCs and VWSs. The files transferred include:

- EDRs
- 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, using 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 121).

Default: .

118  NCC Service Management System Technical Guide
Notes: An example of a destination directory is the directory on a SLC where cmnPushFiles looks for the files to be sent to the SMS.

Example:

- \( P \) \( <\text{dir}> \)

Syntax: \( -P \) \( <\text{dir}> \)

Description: The file prefix to match on.

Type: String

Optionality: Allowed:

Default: Notes:

Example:

- \( S \) \( <\text{sufx}> \)

Syntax: \( -S \) \( <\text{sufx}> \)

Description: The file suffix.

Type: String

Optionality: Allowed:

Default: Notes:

Example:

- \( r \) \( <\text{pref}> \)

Syntax: \( -r \) \( <\text{pref}> \)

Description: The remote directory prefix.

Type: String

Optionality: Optional (default used if not set).

Allowed: Default: Notes: Required if \( -d \) \( <\text{dir}> \) (on page 118) is a relative directory.

Example:

- \( h \) \( <\text{host}> \)

Syntax: \( -h \) \( <\text{host}> \)

Description: The hostname of the remote machine.

Type: String

Optionality: Mandatory

Allowed: Default: Notes: If set, a hostname must be specified.

Example:
-p <port>
Syntax: -p <port>
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.
        -1 Use stdin and stdout.
Default: 2027
Notes:  
Example: 

-s <secs>
Syntax: -s <secs>
Description: The number of seconds for the sleep period.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 
Default: 15
Notes:  
Example: 

-t <bits>
Syntax: -t <bits>
Description: The number of bits per second to start throttling at.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 
Default: 0 (no throttling)
Notes:  
Example: 

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

-x
Syntax: -x
Description: Whether to use hostname-prefixing on remote filenames.
Type: Boolean
### -o <dir>

**Syntax:**

```
-o <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)

**Notes:**

**Example:**

```
-o /tmp
```

### -f <dir>

**Syntax:**

```
-f <dir>
```

**Description:** The retry directory.

**Type:** String

**Optionality:** Optional (default used if not set).

**Allowed:** Cannot be the same as `-d <dir>` (on page 118).

**Default:** null (no retry directory)

**Notes:**

**Example:**

```
-f /tmp/retry
```

### -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

**Notes:**

**Example:**

```
-F
```

### -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 121) option is specified.
Example:

-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 (false) only once, then exit.
        not set Run in daemon mode.
Default: not set
Notes:
Example:

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

-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).
Allowed:
Default: 900
Notes:
Example:

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

\-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  
Notes:  
Example:  

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 prodsmp1.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 SMS 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 SMSSs, 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 path</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 seconds</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 seconds</td>
<td>2</td>
<td>Filter Period - duration between linked-list checks (seconds).</td>
</tr>
<tr>
<td>-n number</td>
<td>5</td>
<td>Filter Number - number of alarm messages allowed within the time period. Allowed values: integers</td>
</tr>
<tr>
<td>-m number</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 SMS 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 node</td>
<td>Direct to the Oracle DB</td>
<td>Rep Node - specify replication requester node.</td>
</tr>
<tr>
<td>-t seconds</td>
<td>1</td>
<td>Commit Interval - maximum interval between committing to the database (seconds).</td>
</tr>
<tr>
<td>-u user/pass</td>
<td>/</td>
<td>Use the supplied Oracle user/password pair.</td>
</tr>
</tbody>
</table>

**Usage examples**

Here is an example 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 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>
</tbody>
</table>
Chapter 6

Background Processes on the SLC

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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.

**logjob.conf**

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`  
**Tip:** Most processes and tools document where their output is written to in their Output topic. |
| 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. |
**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).

  Allowed: 0

  Default: 0

  Notes: This only applies to collection mode 1 (always report).

  Example:
-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).
Allowed:
Default: null
Notes:
Example:

-v
Syntax: -v
Description: Use verbose mode (provide more information while processing).
Type: Boolean
Optionality: Optional (default used if not set).
Allowed:
Default: Do not use verbose mode.
Notes: Usually used for debugging set-up problems.
Example:

-d rows
Syntax: -d rows
Description: The number of rows for dynamic stats table.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: > 1
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.
Example:

-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).
Allowed:
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.
Example:
-f
Syntax: -f
Description: Only do SMS-specific statistic collection.
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: Default: Collect all statistics.
Notes: Only set when smsStatsDaemon is running on an SMS.
The SMF_STATISTICS_EXTN table specifies for each extension statistic whether it is an SMS stat or not. Also the shared memory hash table specifies whether each statistic is an SMS stat. Hence the -F option can do only SMS stats or all stats.
Examples: Uptime of smsMaster (on page 90) is an SMS stat, uptime of updateLoader (on page 136) is not an SMS stat.

Example:

-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).
Allowed: Default: 80
Notes: The dynamic stats table is the shared memory hash table used to contain the statistics information.

Example:

-i
Syntax: -i
Description: Silently ignore mount points that are longer than the -m size (on page 130) limit.
Type: Boolean
Optionality: Optional (not used if not set).
Allowed: Default: Notes: Example:

-S
Syntax: -S
Description: Silently drop statistics where the details field is longer than -m size (on page 130).
Type: Boolean
Optionality: Optional (not used if not set).
Allowed: Default: Notes: Example:
-T
Syntax: -T
Description: Truncate the detail field for statistics that exceed -m size (on page 130).
Type: Boolean
Optionality: Optional (not used if not set).
Allowed:
Default:
Notes:
Example:

-C n
MERSyntax: -C n
Description: Sets the global configuration variable mergeCpuStats to the value of ”n”. This defines whether individual CPU statistics will be output or whether CPU statistics will be summed. For summed statistics, the number of CPUs is also output.
Type: Integer
Optionality: Optional (default used if not set).
Allowed:
  n = 0   output individual CPU statistics
  n = 1   sum CPU statistics
Any other value of n will be ignored, and default behavior will be used
Default: 0 - output individual CPU statistics
Notes: If MERGECPUSTATS config file entry is set, it overrides the default behaviour.
       If the command line switch (-C) is set, then it will override the MERGECPUSTATS config file entry.
Example: -C 1

Parameters for standard mode

This parameters are used with the general parameters when smsStatsDaemon is running in standard mode.

Note: These parameters cannot be used with the Parameters for legacy mode (on page 132).

-u usr/pwd
Syntax: -u usr/pwd
Description: The userid and password to use to log into the SMF database.
Type: String
Optionality: Optional (default used if not set).
Allowed:
Default: /
Notes:
Example:

-r node
Syntax: -r node
Description: The replication node number smsStatsDaemon should use.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 
Default: -1 
Notes: 
Example:

-w 
Syntax: -w 
Description: Do Row Count statistic collection. 
Type: Boolean 
Optionality: Optional (not used if not set). 
Allowed: 
Default: 
Notes: Provides statistics of the number of rows in selected database tables as defined in the SMF_STATISTICS_EXTN table. This type of statistic should only be collected on the primary SMS node.
Example:

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 131).

-c dir 
Syntax: -c dir 
Description: Current statistics file directory. 
Type: String 
Optionality: Optional (default used if not set). 
Allowed: 
Default: /tmp 
Notes: 
Example:

-a dir 
Syntax: -d dir 
Description: Archived statistics file directory. 
Type: String 
Optionality: Optional (default used if not set). 
Allowed: 
Default: /tmp 
Notes: 
Example:

-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).
Allowed: Default: 1800
Notes: Example:

$s$ Kb
Syntax: $-s\ Kb$
Description: The maximum number of Kb a stats file can reach.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Default: 10
Notes: Example:

Failure
If the smsStatsDaemon fails, statistics on that SLC 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.

Statistics shared memory
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.

smsStatsDaemon parameters
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>
</tbody>
</table>
### Name | Type | Description
--- | --- | ---
OPENTIME | unsigned long | The maximum length of time a statistics file should remain open. The range of values is 1-1440 minutes.
MAXSIZE | unsigned long | The maximum size (in Kbytes) a statistics file is allowed to reach.
NOTIFY | 256 characters | 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.
MID | Measurement description record | Specifies a measurement to be made available.
MERGECPUSTATS | 1 character | 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.

### Legacy mode configuration - config file

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
- 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:

```
# 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 NF
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
```
Chapter 6

Where:

<table>
<thead>
<tr>
<th>#</th>
<th>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).</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURRENTDIR</td>
<td>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.</td>
</tr>
<tr>
<td>ARCHIVEDIR</td>
<td>the location a closed statistics file is moved to.</td>
</tr>
<tr>
<td>OPENTIME</td>
<td>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.</td>
</tr>
<tr>
<td>MAXSIZE</td>
<td>The maximum allowable size of a statistics file in Kilobytes.</td>
</tr>
<tr>
<td>Note: If a statistics file becomes overloaded half way through a dump, the entire record will still be written.</td>
<td></td>
</tr>
<tr>
<td>MID</td>
<td>The measurements to retrieve. Refer to the subsection on Measurements</td>
</tr>
</tbody>
</table>

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>
</tbody>
</table>
### Name | Type | Description
---|---|---
DETAIL | 80 characters | Extra data required to measure stat, i.e. process name for process uptime stats. Can be NULL (empty string)
EXTN | 5 characters | The keyword 'EXTN' indicating that this mid line has the extra fields
FUNCTION_NAME | 50 characters | Function within the library (specified at LIB_NAME) to call in order to measure stat e.g. getNodeUptime.
ID | 20 characters | The measurement ID.
KEYWORD | 20 characters | May be required by measurement function. e.g. UPTIME_NODE.
LIB_NAME | 30 characters | Dynamic library to load to get stat measurement function. e.g. libsmsextrastats.so.
PERIOD | unsigned long | The time in seconds between each recording in the output file of this statistic. Value range is 10-31536000 (1yr).

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 SLC. For more information about updateLoaders and replication, see *What is Replication?* (on page 19)

If the SCP data becomes out of sync with the data in the SMF, a resync can be done to ensure the SCP 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 SLC. If this limit is exceeded (the node is out of history) SMS 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 inittab, 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</td>
<td>274</td>
<td>The node number of the updateLoader requesting the resync. Causes the updateLoader to re-synchronise with the smsMaster.</td>
</tr>
<tr>
<td>node_number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-resync</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Failure

If the updateLoader is not working, updates from the SMS to the SCP database will be unsuccessful. The SLC will continue to run on the last configuration successfully loaded from the SMS.

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 7
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
- cmnSU
- compareNode
- comparisonServer
- inetCompareServer
- infoDisplayer
- inputBootstrap
- repConfigWrite
- resyncServer
- smsCompareResyncClient
- smsCompareResyncServer
- smsDumpRepConfig
- smsLogTest
- smsManualRequester
- smsProcessCdr
- smsRecordStatistic
- smsStatsQuery
- startMerge

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
```

**cmnSU**

**Purpose**

cmnSU replaces the built in Solaris 11 "su" command for NCC processes run from initab on Solaris 11 environments only. Run as root, it will provide a login shell to the specified user.

**Configuration**

cmnSU accepts the following command line options.

**Usage:**

```
/cmnu - username [arg...]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td></td>
<td>Provide a login shell. Required.</td>
</tr>
<tr>
<td>username</td>
<td></td>
<td>The user to become. Required.</td>
</tr>
<tr>
<td>[arg...]</td>
<td></td>
<td>The rest of the arguments to the shell</td>
</tr>
</tbody>
</table>

**compareNode**

**Purpose**

This command can be used to initiate a full database comparison of an SCP database with the definitive copy in the SMF database.

This is used to ensure that an SCP 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 SLC 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 SLC machine.
**Configuration**

compareNode accepts the following command line options.

**Usage:**

```bash
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.) <strong>Note:</strong> This can be an infMaster.</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>

**Example:** This text shows the common usage of compareNode being run on the superior master in a node comparison.

```bash
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`.

**comparisonServer**

**Purpose**

comparisonServer is a shell script which starts node comparisons between data in two different nodes. It is started by an SMS in response to a database comparison request.

**Configuration**

comparisonServer accepts the following command line options.

**Usage:**

```bash
comparisonServer node_to_compare address port
```
The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>node_to_compare</td>
<td></td>
<td></td>
</tr>
<tr>
<td>address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>port</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Output

comparisonServer writes error messages to the system messages file, and also writes additional output to /IN/service_packages/SMS/tmp/comparisonServer.log.

inetCompareServer

Purpose

inetCompareServer is a shell script which is run by the Replication Check screens. It uses the report configuration information from the Replication Check screen to start a node comparison (which is performed by smsCompareResyncServer). It should not be necessary to run this script by hand.

Output

inetCompareServer writes to syslog and also logs additional information (including raw Replication Check report data) to /IN/service_packages/SMS/tmp/inetCompareServer.log.

Example: This text shows an example of a report from the inetCompareServer.

Copyright (c) 2002, Oracle. Contact Oracle at support@oracle.co.nz

Input delivered through standard input. Please consult the SMS administration guide for more information on this software. For command line options, pass '-h' as the only option to this program.

Awaiting server control information...

Input accepted. Now running server.
Mar 4 05:01:49 smsCompareResyncServer(28781) NOTICE: Beginning comparison for node 301.
COM: Fri Mar 4 05:01:50 2005: Node 301, started processing 186 SMS and 186 SCP records.
COM: Fri Mar 4 05:01:50 2005: Node 301, table ACS_CALL_PLAN, started processing 186 SMS and 186 SCP records.
COM: Fri Mar 4 05:01:51 2005: Node 301, table ACS_CALL_PLAN, group ACS_CALL_PLAN, started processing 186 SMS and 186 SCP records.
COM: Fri Mar 4 05:01:52 2005: Node 301, table ACS_CALL_PLAN, group ACS_CALL_PLAN, finished processing 186 of 186 SMS and 186 of 186 SCP records, 0 discrepancy found in group.
COM: Fri Mar 4 05:01:52 2005: Node 301, table ACS_CALL_PLAN, finished processing 186 of 186 SMS and 186 of 186 SCP records, 0 discrepancy found in table.
COM: Fri Mar 4 05:01:54 2005: Node 301, finished processing 186 of 186 SMS and 186 SCP of 186 records, 0 discrepancy found in node.
Mar 4 05:01:54 smsCompareResyncServer(28781) NOTICE: Ending comparison for node 301.
Mar 4 05:01:54 smsCompareResyncServer(28781) NOTICE: Comparison was successful for node 301.
Started writing index HTML file for reports.
Finished writing index HTML file for reports.

**infoDisplayer**

**Purpose**

infoDisplayer is an executable which can be used to display update request information results.

**Configuration**

infoDisplayer supports the following command-line options:

**Usage:**

```
infoDisplayer [-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 infoDisplayer will timeout</td>
</tr>
</tbody>
</table>

**Output**

**Examples:**

```
bash-2.05$ ./infoDisplayer -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 infoDisplayer(6589) NOTICE: Master Controller `./infoDisplayer' 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</td>
<td>none</td>
<td>The replication node ID for which the configuration file is produced. (Required.)</td>
</tr>
<tr>
<td>--node-id</td>
<td></td>
<td>Allowed values: integers, (any signed number of reasonable value, usually in decimal/octal/hex).</td>
</tr>
<tr>
<td>-c</td>
<td>/IN/service_packages/</td>
<td>Name of the replication configuration file to use. (Optional.)</td>
</tr>
<tr>
<td>--config-file</td>
<td>SMS/etc/replication.config</td>
<td>Allowed values: string</td>
</tr>
<tr>
<td>-a</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.</td>
</tr>
<tr>
<td>--address</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>-p</td>
<td>none</td>
<td>The port number to connect to at the given node. (Optional.)</td>
</tr>
<tr>
<td>--port</td>
<td></td>
<td>Note: This can only be used when the address is also specified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: integers, (any signed number of reasonable value, usually in decimal/octal/hex).</td>
</tr>
</tbody>
</table>
Parameter | Default | Description
--- | --- | ---
\-r | false when missing | 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. **Allowed values:**
- 1, on, yes, true
- 0, off, no, false

\-u | smf/smf | The Oracle database connection string (userid/password). (Optional.) **Allowed values:** string

\-e | false when missing | Restrict range of rows to resync. If enhanced recovery mode is possible, the smsMaster process sets this option. **Allowed values:**
- 1, on, yes, true
- 0, off, no, false

\-h | false when missing | Shows the help for this binary. **Allowed values:**
- 1, on, yes, true
- 0, off, no, false

\-b | false when missing | Prints out program build information of the binary. **Allowed values:**
- 1, on, yes, true
- 0, off, no, false

**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, \-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 using the SMS Java screens.

For more information, see:
- replication.config File (on page 37)
- smsDumpRepConfig (on page 159)
- SMS User's Guide
Chapter 7

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 SMS 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: <code>smf/smf</code></td>
<td></td>
</tr>
<tr>
<td>-output</td>
<td><code>/repConfigNNNNN</code> (Where <code>NNNNN</code> is a version number that counts for the number of times the file has generated OK.)</td>
<td>The output path and filename for the replication.config file. (Optional.) Example: <code>/IN/service_packages/SMS/etc/replication.config</code></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 SCP 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>inf_node</td>
<td></td>
<td>The node with the database which will be updated.</td>
</tr>
<tr>
<td>address</td>
<td></td>
<td>The ip address or hostname of the node which will be updated.</td>
</tr>
<tr>
<td>port</td>
<td></td>
<td>The port number on the node which will be updated.</td>
</tr>
<tr>
<td>enhanced_recover</td>
<td>off</td>
<td>If set to on, the number of rows in the inferior database will not be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>counted during the resync.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- on, 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 SCP during replication on a clustered install. It also performs database resynchronizations and comparisons on the inferior node during replication checks.

This process is not intended to be be started manually. It is installed on the SLC.

**Configuration**

smsCompareResyncClient accepts the following command line options.

**Usage:**

smsCompareResyncClient -n int [-u usr/pwd] [-h] [-i int] [-p port] [-o seconds]


The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-n</td>
<td></td>
<td>The node number of the client. (Required.)</td>
</tr>
<tr>
<td>--node-id</td>
<td></td>
<td><strong>Allowed values:</strong> integer (any signed number of reasonable value, usually in decimal/octal/hex)</td>
</tr>
<tr>
<td>-u</td>
<td>smf/smf</td>
<td>The Oracle database connection string. (Optional.)</td>
</tr>
<tr>
<td>--oracle-user</td>
<td></td>
<td><strong>Allowed values:</strong> string</td>
</tr>
<tr>
<td>-h</td>
<td>false</td>
<td>Prints this help screen. (Optional.)</td>
</tr>
<tr>
<td>--help</td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1, on, yes, true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0, off, no, false</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------</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></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0, off, no, false</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></td>
<td>Allowed values: integers (any signed number of reasonable value, usually in decimal/octal/hex)</td>
</tr>
<tr>
<td>-p --port</td>
<td></td>
<td>The port to listen on for connections from the server. (Optional.)</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>-o --tcp-timeout</td>
<td>-1</td>
<td>Timeout (in seconds) on TCP connection. (Optional.)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allowed values: 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>Allowed values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1, on, yes, true</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>Allowed values: integers (any signed number of reasonable value, usually in decimal/octal/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>Allowed values: integers (any signed number of reasonable value, usually in decimal/octal/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>Allowed values: integers (any signed number of reasonable value, usually in decimal/octal/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>Allowed values: integers (any signed number of reasonable value, usually in decimal/octal/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>Allowed values: 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>Allowed values: integers (any signed number of reasonable value, usually in decimal/octal/hex)</td>
</tr>
</tbody>
</table>
### 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
- 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 SMS.

#### Configuration

smsCompareResyncServer accepts the following command line options.

**Usage:**

<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 terms of records of the buffer size for reading to the database. (Optional.)&lt;br&gt;<strong>Allowed values:</strong> integers (any signed number of reasonable value, usually in decimal/octal/hex)</td>
</tr>
<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.)&lt;br&gt;<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.)&lt;br&gt;<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.&lt;br&gt;<strong>Allowed values:</strong>&lt;br&gt;1, on, yes, true&lt;br&gt;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.&lt;br&gt;<strong>Allowed values:</strong>&lt;br&gt;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

```
--dump-core-instead-of-exception --max-buffer-size=size [--
dont-count-rows] [--inform-master] [--database-read-buffer-size=size] [--cancel-on-
eof] [--use-ip=int] [--report-directory=base_dir] [--tcp-timeout] [--input-
file=config_file_name] [--oracle-user=user] [--build-info] [--help] [--long-raw-
size=max_size]
```

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><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1, on, yes, true</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><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- integers (any signed number of reasonable value, usually in decimal/octal/hex).</td>
</tr>
<tr>
<td>-d</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>--dont-count-rows</td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1, on, yes, true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0, off, no, false</td>
</tr>
<tr>
<td>-m</td>
<td>false when missing</td>
<td>If performing a resync, inform the smsMaster.</td>
</tr>
<tr>
<td>--inform-master</td>
<td></td>
<td><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1, on, yes, true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0, off, no, false</td>
</tr>
<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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>smsCompResyncClient. (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/octal/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><strong>Allowed values:</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 1, on, yes, true</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- 0, off, no, false</td>
</tr>
<tr>
<td>Parameter</td>
<td>Default</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</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. <strong>Allowed values:</strong> integers (any signed number of reasonable value, usually in decimal/octal/hex). <strong>Examples:</strong> If --use-ip = 2, the server will use the second IP address listed.</td>
</tr>
<tr>
<td>-r</td>
<td>/IN/html/output/SMS/</td>
<td>The base directory to create and store final reports in. (Optional.) <strong>Allowed values:</strong> string.</td>
</tr>
<tr>
<td>--report-directory</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-o</td>
<td>0</td>
<td>Timeout (in seconds) on TCP connection. (Optional.) Zero = no timeout. <strong>Allowed values:</strong> integers (any signed number of reasonable value, usually in decimal/octal/hex).</td>
</tr>
<tr>
<td>--tcp-timeout</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-i</td>
<td></td>
<td>Input is expected on the standard input stream. Contains the name of a configuration file as input information for performing a resync/compare. (Optional.) See Input file for details. <strong>Allowed values:</strong> string</td>
</tr>
<tr>
<td>--input-file</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-u</td>
<td>smf/smf</td>
<td>The Oracle database connection string. (Optional.) <strong>Allowed values:</strong> string</td>
</tr>
<tr>
<td>--oracle-user</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-b</td>
<td>false when missing</td>
<td>Prints out program build information then exits. <strong>Allowed values:</strong> 1, on, yes, true 0, off, no, false</td>
</tr>
<tr>
<td>--build-info</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-h</td>
<td>false when missing</td>
<td>Prints this help screen. <strong>Allowed values:</strong> 1, on, yes, true 0, off, no, false</td>
</tr>
<tr>
<td>--help</td>
<td></td>
<td></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. <strong>Allowed values:</strong> 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.
Chapter 7

Replication

Syntax:  Replication = \{list_of_replication_parameters\}
Description: The Replication section lists the required replication parameters.
Type:  List
Optionality: Mandatory
Allowed: 
Default: none
Notes: 
Example: 

perform

Syntax:  perform = "action"
Description: The action to be taken by smsCompareResyncServer.
Type:  Boolean
Optionality: Mandatory
Allowed: 
• resync
• compare
Default: none
Notes: 
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 synchronized.
Type:  Integer
Optionality: Optional (default used if not set).
Allowed: 
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)
Notes: 
Example: produce-final-reports = false
**report-directory**

**Syntax:**

\[
\text{report-directory} = \text{"dir"}
\]

**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

**Notes:**
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).

**Allowed:**
No progress reports are provided.

**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 152) is reached.

**Type:**
Boolean

**Optionality:**
Optional (default used if not set).

**Allowed:**
true, false

**Default:**
false (do not stop)

**Notes:**
Example:

```
stop-on-limit = false
```

**view**

**Syntax:**

```
view = {}
```

**Description:**
Describes the nodes, tables and groups for the replication action.

**Type:**
List

**Optionality:**
Mandatory
Chapter 7

Allowed: none
Default: none
Notes: 
Example: 

Node
Syntax: Node 
   id = number
   address = [ "string", "string", ... ]

Description: A list of nodes for the replication action to work on.
Type: Array
Optionality: Mandatory
Allowed: 
Default: none
Notes: 
Example: 

id
Syntax: id = id
Description: The ID of the node to be used.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: Values from replication node configuration
Default: 
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

address
Syntax: address = [ "string", "string", ... ]
Description: An array of IP addresses and port number(s) for this node ID.
Type: Array
Optionality: Mandatory
Allowed: 
Default: none
Notes: 
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
Allowed: 
Default: none
Notes: 
Example: 

```
tables = [
  table = "<str>"
  key-columns = [
    "str", "str", ...
  ]
  other-columns = [
    "str", "str", ...
  ]
]
```

Example: table = "TEST_REP"

**groups-cover-table-on-scp**

Syntax: 

```
groups-cover-table-on-scp = true|false
```

Description: 
Type: Boolean
Optionality: Optional (default used if not set).
Allowed: true, false
Default: false
Notes: 
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:
\[
\text{key-columns} = [
    \"NUMBER_3\"
]
\]

other-columns

Syntax: \[
\text{other-columns} = [\n    \"str\", \"str\", ...
\]
\]
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:
\[
\text{other-columns} = [
    \"VARCHAR_1","LONG_RAW_2","CHAR_4","DATE_5\"
]
\]

Groups

Syntax: \[
\text{groups} = [\n    \{
        \text{group} = <str>, 
        \text{table} = <str>, 
        \text{ranges} = [\n            <rangeData>, <rangeData>, ...
        ],
        \text{nodes} = [\n            <int>, <int>, ...
        ]
    \}
]
\]
Description: An array of groups associated with this operation.
Type: Array
Optionality: Mandatory
Allowed: 
Default: none
Notes:
Example:
group

Syntax: \[
\text{group} = "<str>"
\]
Description: The name associated with a node.
Type: String
Optionality: Mandatory
Allowed: Any character
Default: none
Notes:
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
Notes:
Example: table = "TEST_REP"

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
Allowed: 
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.
Chapter 7

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 = 

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"
                ]
            }
        ]
    }
]
Output
/nsmsCompareResyncServer writes error messages to the system messages file.
smsCompareResyncServer writes replication checks and database comparisons to the
/IN/html/output/SMS/compare/inferior_node_number directory.

**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 37).

**Configuration**

The smsDumpRepConfig supports the following command-line options:

**Usage:**

```
smsDumpRepConfig -f filename [-v]
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-f</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...

---

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:

```bash
./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>name</td>
<td>none</td>
<td>The subsystem name/identifier. (Required.)</td>
</tr>
</tbody>
</table>
### Chapter 7, Tools and Utilities

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
</table>
| severity        | none    | The severity value. (Required.)
|                 |         | Allowed values: |
|                 |         | N or 0 [Notice] |
|                 |         | W or 1 [Warning] |
|                 |         | E or 2 [Error] |
|                 |         | C or 3 [Critical] |
|                 |         | - or 4 [clear] |
| message         | none    | The message to log. (Required.) |
| [copies]        | 1       | Number of times to generate the alarm. (Optional.) |
| [{alarm_type_id}] | none | Optional Alarm Type ID associated with the message. This must be up to a 9-digit number between braces (for example, {123456789}) |

**Examples:**

```bash
./smsLogTest smsAlarmRelay %d "Failed to connect to Oracle" 4
n {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></td>
<td>the ID of the node (Optional)</td>
</tr>
</tbody>
</table>

**Output**

`smsManualRequester` displays the output to local terminal.

**Examples:**

```bash
./smsManualRequester -nodeid 999
```
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 ##):##
nInitialiseNode: 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: 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)
- 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 86).

**Configuration**

smsProcessCdr accepts the following command line parameters.

**Usage:**

```
```

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t edr_format</td>
<td>none</td>
<td>The filename of the EDR format file.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> No format conversion is performed by default. The formatting</td>
</tr>
<tr>
<td></td>
<td></td>
<td>file supports the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Fields</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Literal strings</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Functions</td>
</tr>
<tr>
<td>-d in_dir</td>
<td>none</td>
<td>The directory to read EDRs from.</td>
</tr>
<tr>
<td>-D out_dir</td>
<td>none</td>
<td>The directory to write processed EDRs to.</td>
</tr>
<tr>
<td>-s in_suffix</td>
<td>none</td>
<td>The suffix that input EDR files must match (these are stripped from the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output file name).</td>
</tr>
<tr>
<td>-p in_prefix</td>
<td>none</td>
<td>The prefix that input EDR files must match (these are stripped from the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>output file name).</td>
</tr>
<tr>
<td>-S out_suffix</td>
<td>none</td>
<td>The suffix to add to output EDR files.</td>
</tr>
<tr>
<td>-P out_prefix</td>
<td>none</td>
<td>The prefix to add to output EDR files.</td>
</tr>
<tr>
<td>-u usr/pwd</td>
<td>/</td>
<td>The Oracle user and password to use.</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note:</strong> smsProcessCdr will only attempt to connect to the database if</td>
</tr>
<tr>
<td></td>
<td></td>
<td>the EDR format file contains functions that require it.</td>
</tr>
<tr>
<td>-l tz</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>
<tr>
<td>-b</td>
<td>none</td>
<td>Allows blank header tag values. Treats non-existent header tags as blank</td>
</tr>
<tr>
<td></td>
<td></td>
<td>value.</td>
</tr>
</tbody>
</table>

**Example 1**

The following command would:

- Use `/IN/service_packages/SMS/bin/cdrFormat.fmt` as the EDR format file
- 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 named `/tmp/processedCdrs/ACS_20010831120012.out`. 
Example 2

The following command:

```
smProcessCdr -t /IN/service_packages/SMS/bin/mobifone.fmt -d
/IN/service_packages/SMS/cdr/received -D /tmp/processedCdrs -p scp2_acs -s .cdr -P
ACS -S .out -u smf/smff
```

Would cause the following file to be parsed as the EDR format file

/IN/service_packages/SMS/bin/mobifone.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.

The 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.

The valid field specifiers are:

- Header tags
- Standard fields
- Special fields
- Strings
- Functions
- Format characters

Header tags

A HEADER tag may be specified in the format file passed in providing a single header line at the top of processed output files.

The header appears once and contains all HEADER tag values concatenated and space separated.

Through the use of a `--b` option passed in to smsProcessCdr at runtime blank values are allowed. When any tags are missing their respective value will be set blank rather than exiting on error. Underscores are allowed by default with no extra settings.

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 a EDR which does not occur in the input as a `tag=value` pair.
The only example of this is the EDR application name field, which always occurs as the first element in a EDR.

Placing the field \text{<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 \text{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:

\[ \text{function\_name ( function\_parameters )} \]

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
- Functions

**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.

\[ \text{EQUALS ( expr1 , expr2 )} \]

**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".

\[ \text{COND ( ( EQUALS ( APPLICATION , "ACS" ) , "ACS Service" ) , ( TRUE , "Unknown Service" ) )} \]

**PREFIX function**

The PREFIX function evaluates to TRUE if expr2 is a prefix of expr1. Otherwise it evaluates to FALSE.

\[ \text{PREFIX ( expr1 , expr2 )} \]
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 ) )

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 ( ( bexpr1 , expr1 ) [ , ( bexpr2 , expr2 ) . . . , ( bexprn , exprn ) ] )

Examples:

In either of these two examples, the function evaluates to:

- Pizza Shed if the service number is 0800101101
- Pay Service if the service number starts with 0900
- Unknown in all other cases

COND ( ( EQUALS ( SN , "0800101101" ) , "Pizza Shed" ) , ( TRUE , ( COND ( ( PREFIX ( SN , "0900" ) , "Pay Service" ) , ( TRUE , "Unknown" ) )

COND ( ( EQUALS ( SN , "0800101101" ) , "Pizza Shed" ) , ( 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 ( "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 ( ( EQUALS ( LANGUAGEID ( "English" , LGID ) ) , "1" ) , ( EQUALS ( LANGUAGEID ( "French" , LGID ) ) , "2" ) , ( EQUALS ( LANGUAGEID ( "German" , LGID ) ) , "3" ) )
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.

\[
\text{ROUND ( expr )}
\]

Examples:
This example evaluates to 2.
\[
\text{ROUND ( "2.1" )}
\]
This example evaluates to 3.
\[
\text{ROUND ( "2.6" )}
\]
If CCET evaluates to 12.3, this example evaluates to 12.
\[
\text{ROUND ( CCET )}
\]

SUBSTR function
The SUBSTR function extracts a substring from a given expression. The parameters:
- \text{expr} is the source expression
- \text{start} is an integer indicating where the substring should start (counting starts at zero)
- \text{length} is an integer indicating how many characters should be read.
If \text{start} is greater than the length of \text{expr}, then SUBSTR returns an empty string. If the specified \text{start} and \text{length} would cause SUBSTR to read off the end of the input expression, then SUBSTR returns the maximum number of characters it could read.

\[
\text{SUBSTR ( expr, start, length )}
\]

Examples:
This example evaluates to "the".
\[
\text{SUBSTR ( "the happy elephant" , 0, 3 )}
\]
This example evaluates to "e ha".
\[
\text{SUBSTR ( "the happy elephant" , 2, 4 )}
\]
This example evaluates to an empty string.
\[
\text{SUBSTR ( "the happy elephant" , 50, 4 )}
\]
This example evaluates to "appy elephant".
\[
\text{SUBSTR ( "the happy elephant" , 5, 40 )}
\]

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>
<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:

```
<APPLICATION> "," SN "," CLI \n
Input CDR 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:

```
// 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)) )
// end of line indicator:
\n \r
```

Input CDR 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 00321321
ACS 0 0 9 0123123
VPN 0 0 -2 00123123
CCS 0 0 0 00321321
```

**File Format example 3**

Another complicated example using a header, comments, special fields, and a function.

Fields are space delimited, records are terminated with a newline and a carriage return character.

Format file:

```
HEADER ( "ONE TWO" )
//ROUND ( "6.1" )

<APPLICATION> " "
ROUND ( CCET ) " "
HEADER ( "THREE" )
COND ( (EQUALS(<APPLICATION>, "CCS"), CONCAT("00", CA)),
{TRUE, CONCAT("00", TN)) })
// end of line indicator:
\n \r
```

Input CDR 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
```
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

Purpose

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.

Location

The smsRecordStatistic process is located on the SLC in the \service_packages\SMS\bin directory.

Configuration

smsRecordStatistic supports the following command-line options:

Usage:
smsRecordStatistic [application] [statistic] [value]

The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[application]</td>
<td>The name of the application for the statistic. (Optional)</td>
</tr>
<tr>
<td>[statistic]</td>
<td>The name of the statistic to record. (Optional)</td>
</tr>
<tr>
<td>[value]</td>
<td>Adds the given delta value to the statistic. (Optional.)</td>
</tr>
</tbody>
</table>

Output

The statistic named when running the script will be updated in the database.

smsStatsQuery

About smsStatsQuery

The smsStatsQuery utility enables you to directly query statistics generated on the Voucher and Wallet Server (VWS) and Service Logic Controller (SLC) before the statistics are replicated to the Service Management System (SMS). You use this utility for system monitoring.
Tip: You can view statistics that have been replicated to the SMS node by using the statistical viewing feature in the SMS user interface (UI). For more information, see SMS User's Guide.

The smsStatsQuery utility is located in the following directories:

- /IN/service_packages/BE/bin on the VWS nodes.
- /IN/service_packages/SMS/bin on the SLC nodes.

You can either supply a single query string as input to smsStatsQuery, or you can supply a text file containing a list of query strings as input.

Usage:

```bash
smsStatsQuery [options] "stats_query"
smsStatsQuery [options] -f "queryfile"
```

Where:

- `options` is a space-separated list of optional parameters. The available options include options for the standard bc (binary calculator) Solaris utility, that is accessed by smsStatsQuery to apply calculations to the statistics results. The optional parameters and some typical bc options that you might want to set are listed in the [Optional Parameters Table](#) on page 171 below.

  **Note:** You can display a full list of bc options by entering `man bc` at the UNIX prompt.

- `stats_query` is a string that identifies the statistics to query. To retrieve multiple statistics, specify a space-separated list of the statistics you want in the query string. You can also include a mathematical formula in the query to perform calculations on the retrieved statistics and return a single value.

  See [Specifying the Statistics to Query](#) on page 170 for information about specifying the `stats_query` string.

- `queryfile` is the name of a text file that contains a list of `stats_query` strings.

  **Note:** Specify either `stats_query` or `queryfile`, but not both.

Examples

In the following examples, the statistics to query are specified in a query string:

```bash
./smsStatsQuery "Acs_Service.elapsedTime"
./smsStatsQuery "Acs_Service.CALLS_INITIATED Acs_Service.ANNOUNCEMENTS_PLAYED"
```

In the following example, the statistics to query are specified in a text file:

```bash
./smsStatsQuery -f "queryFile.txt"
```

**Note:** The statistics on the VWS and SLC nodes are collected by smsStatsDaemon for replication to the SMS node. If you enter a query for a statistic that is not currently held by smsStatsDaemon then the smsStatsQuery utility returns an error. You can check which statistics are currently held by smsStatsDaemon by entering the following command:

```bash
./smsStatsQuery -l
```

For more information about smsStatsDaemon, see [smsStatsDaemon](#) on page 128.

Specifying the Statistics to Query

To specify one or more statistics in the `stats_query` string, use the following syntax:

```bash
application.statistic[.detail] [application.statistic[.detail]]
```

Where:

- `application` is the name of the application or service that generated the statistic, such as `Acs_Service`.
- `statistic` is the name of the statistic to query, such as `elapsedTime`.
- `detail` (optional field) is the name of a detail field associated with the specified statistic. **Note:** Not all statistics have detail fields.
For example, the following queries retrieve statistics for the ACS service:

```
./smsStatsQuery "Acs_Service.elapsedTime"
./smsStatsQuery "Acs_Service.CALLS_INITIATED Acs_Service.ANNOUNCEMENTS_PLAYED"
```

To query a statistic that contains a space in any of its attribute names, you must use double square brackets, `[[" and "]]`, to enclose the statistic specification in the `stats_query` string.

For example, the following queries include the "PrePaid Success" statistic in the `stats_query` string:

```
./smsStatsQuery "[[Ccs_Service.PrePaid Success]]"
./smsStatsQuery "Acs_Service.CALLS_INITIATED [[Ccs_Service.PrePaid Success]] Acs_Service.ANNOUNCEMENTS_PLAYED"
```

To specify a formula in the `stats_query` string, use the following syntax:

```
[factor]statistic[[factor][statistic]]
```

Where:
- `factor` is a combination of a constant and an operator, or just an operator, that is applied to the statistic.
- `statistic` is a statistic specified as: `application.statistic.detail`.

For example:

```
./smsStatsQuery "10*Acs_Service.ANNOUNCEMENTS_PLAYED/Acs_Service.CALLS_INITIATED"
```

**Note:** You may not retrieve a list of statistics if you include a formula in the `stats_query` string.

### Optional Parameters Table

This table describes the optional parameters for `smsStatsQuery`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-h</code></td>
<td>Lists usage information for the <code>smsStatsQuery</code> utility.</td>
</tr>
<tr>
<td><code>-l</code></td>
<td>Lists the contents of statistics currently held by <code>smsStatsDaemon</code>.</td>
</tr>
<tr>
<td><code>-t secs</code></td>
<td>(bc option) Calculates the average rate a statistic is used based on two readings of the statistic, where the second reading is taken <code>secs</code> seconds after the first reading. The formula used to calculate the average rate a statistic is used is: <code>(value of reading 2 - value of reading 1)/ secs</code>.</td>
</tr>
<tr>
<td><code>-p precision</code></td>
<td>(bc option) Specifies the number of decimal places to display for the statistics value output.</td>
</tr>
</tbody>
</table>

### 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 an superior master by merging it with an inferior master.

**Configuration**

The `startMerge` supports the following command-line options:

**Usage:**

```
startMerge [-from nodenum] [-to nodenum]
```
The available parameters are:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-from nodenum</td>
<td>none</td>
<td>Node number of the inferior master to merge from.</td>
</tr>
<tr>
<td>-to nodenum</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.
Overview

Introduction

This chapter explains SMS reporting functionality.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reports Database Tables</td>
<td>173</td>
</tr>
<tr>
<td>Installing a Report Script</td>
<td>174</td>
</tr>
<tr>
<td>Report Script Worked Example</td>
<td>176</td>
</tr>
<tr>
<td>Database Auditing</td>
<td>180</td>
</tr>
</tbody>
</table>

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:

- Service reports, to be installed at the time of service installation
- 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>
<tr>
<td>DESCRIPTION</td>
<td>Help text.</td>
</tr>
<tr>
<td>TYPE</td>
<td>The type – INT, STRING, DATE, and so on. (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 SMS 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;  

<table>
<thead>
<tr>
<th>APP_ID</th>
<th>APPLICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SMS</td>
</tr>
<tr>
<td>4</td>
<td>SYSTEM</td>
</tr>
<tr>
<td>2</td>
<td>Acs_Service</td>
</tr>
</tbody>
</table>

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".

2

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.

3

A report must one of the following:

- Have a *sql* extension. In this case, it will be executed using sqlplus.
- 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.

Examples:

For a *sql* file:

`sqlplus script-name output-file [user-parameters]`

For an executable file without a *sql* extension

`script-name output-file [user-parameters]`

Exit status of report scripts are defined by the following:

- 0 = okay
- > = not okay. (Unix style)
- <0 = undefined

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.

The user may request cancelation of a script, in which case it will be sent a SIGTERM. Scripts should not ignore SIGTERM.

If the report spawns children, it should implement a SIGTERM handler to dispose the children, in case the user cancels a report.

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> 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.
<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>The script itself must be placed into <code>/IN/service_packages/SMS/input/application-name/category/script-file</code>.</td>
</tr>
</tbody>
</table>
| 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**

**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 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 SMS in the correct location.
5. Insert entries into the REPORT tables in the SMF database.

**Example report script**

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>
</table>
| 1    | The details for this example are:  
  Application: SMS (ID is 1)  
  Category: “Errors”  
  Name: “Program Errors”  
  Script File: `program_errors.sql`  
  Application Part: 1805 (new part)  
  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.  
  We are free to choose the category; we have chosen the “Errors” category.  
  We are free to choose a unique script name within this application and category; we have chosen `program_errors.sql`. 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. |
Step | Action
--- | ---
2 | The report takes three user parameters:
   Num Hours: Integer in range 1..999, default is 24
   Program Prefix: String length 0..20 characters
   Category: One of FATAL, SERIOUS, WARNING, INFORMATIONAL

**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.

```sql
/*----------------------------------------------
* File: program_errors.sql
* Updates:
* Parameters:  &1 Output file, determined by reports daemon &2 Hours back &3 Program prefix &4 Severity (FATAL,SERIOUS,WARNING,INFORMATIONAL)
*---------------------------------------------------------------------*/
-- #ident "@(#)$Id: telephony_errors.sql,v 1.4 1999/02/25 22:10:29 rhwang Exp $"
-- so we won't print to stdout.
set termout off
set verify off
-- we are going to access the sms_program_errors table.
-- set the column titles.
column program 'Program' format a20
column error_code 'Error Code' format a16
column node_name 'Node Name' format a20
column severity 'Error Severity'
set linesize 80
set pagesize 2100
spool &1
-- now set the title at the top of the page.
ttitle center 'Recent Telephony Errors for Application &3 Severity &4' skip 1 -
center 'PROGRAM TELEPHONY ERRORS' -
center ============================ skip 1 -
RIGHT 'PAGE:' FORMAT 999 SQL.PNO SKIP 2
```
### Chapter 8

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<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 smf_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>
<tr>
<td></td>
<td>quit</td>
</tr>
</tbody>
</table>

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 SMS, 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;
```
Step | Action
--- | ---
4 | As part of the installation package, ensure that the file, `program_errors.sql` is installed into the correct destination location, for example, `/IN/service_packages/SMS/input/SMS/Errors/program_errors.sql`. The `smf_oper` 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 `smf_oper` also has execute access to this file.
5 | The final task is to notify the SMS about the script, to make it visible. In this example, the report and the three user parameters to be collected are defined.

**Note:** 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.

```sql
/*
 * 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');

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,WARNI

```
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>usr/pwd</td>
<td></td>
<td>The user and password combination to be used to log into the SMF. (Required.)</td>
</tr>
<tr>
<td>start_date</td>
<td></td>
<td>The time and date the query will start reporting on. The format is yyyymmddhh24mms. (Optional.)</td>
</tr>
<tr>
<td>end_date</td>
<td></td>
<td>The time and date the query will stop reporting on. The format is yyyymmddhh24mms. (Optional.)</td>
</tr>
<tr>
<td>db_user</td>
<td></td>
<td>The userid for the database user which made the changes to the database. (Optional.)</td>
</tr>
<tr>
<td>table</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 stdout.

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
SU,20050308225808,192168007165,SMF_USER,ADMIN_TRAINING1_EX2,Student Account,Student 1,0,Locked for testing,31,20050406000000,LANGUAGE=ENGLISH
SU,20050308225808,192168007165,SMF_USER,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

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
Overview

Introduction

This chapter explains the important processes on each of the server components in the NCC, and a number of example troubleshooting methods which will help aid the troubleshooting process before raising a support ticket.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Troubleshooting Procedures</td>
<td>183</td>
</tr>
<tr>
<td>Possible Problems</td>
<td>184</td>
</tr>
<tr>
<td>Index Defragmentation</td>
<td>186</td>
</tr>
</tbody>
</table>

Common Troubleshooting Procedures

Introduction

Refer to *NCC System Administrator’s Guide* for troubleshooting procedures common to all NCC components.

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 webserver.

**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 SLC.

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 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

Introduction

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 behavior you are experiencing.

SMS Java screens will not start

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 SMS) 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.jnlp* file is correctly configured.  
- Ensure that the SMS console is able to resolve host names into IP addresses. |

Java help screen grayed out

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 <strong>Control Panel</strong>.</td>
</tr>
</tbody>
</table>
### Replication is failing

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 37).</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>
comparisonServer is failing

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

Description

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 fragmentation_install.sh must first be installed. This will install the stored procedure sms_defrag_rep_iot, and schedule a job to run it every 10 minutes.

The following tables are affected:

- REP_ORA_EVENT
- REP_ORA_RENUMBERED

Before you begin

The process for installing the defragmentation script varies depending on the Oracle configuration available on the SMS. 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 32 KB block size should be set to 32 MB 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
buffer_pool_keep
buffer_pool_recycle}
Overview

Introduction

This chapter explains the pre-installation configuration requirements of the application.

In this chapter

This chapter contains the following topics.

SMS Client Specifications 189
Preparing the System 190
Database Timezone and Backups 191
Starting Oracle Automatically on Reboot 192

SMS Client Specifications

Specifications

This topic provides the specifications of SMS.

Network

The minimum requirements of network bandwidth for acceptable normal response times are as follows:

<table>
<thead>
<tr>
<th>Number of Users</th>
<th>Minimum Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-5</td>
<td>512 KB</td>
</tr>
<tr>
<td>6-15</td>
<td>1 MB</td>
</tr>
<tr>
<td>16 +</td>
<td>LAN connection (at least 25% available resource of 10 MB)</td>
</tr>
</tbody>
</table>

Memory

The NCC screens are written to optimize data interaction. As a result, it is necessary to cache data in such a way as to reduce redundant data retrieval. This means that heavy usage can lead to the requirement for a large amount of memory to be available on the client machine running the screens. The recommended memory installed on the client machine is 256MB minimum with 512 MB preferred, especially with machine running Windows XP.

This table shows the minimum client resources required.

<table>
<thead>
<tr>
<th>RAM</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>256 MB</td>
<td>800 MHz</td>
</tr>
</tbody>
</table>
This table shows the recommended client resources required.

<table>
<thead>
<tr>
<th>RAM</th>
<th>CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>512 MB</td>
<td>1.2 GHz</td>
</tr>
</tbody>
</table>

**Response times**

This table shows typical response time.

<table>
<thead>
<tr>
<th>GUI Action</th>
<th>Response Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startup to Login dialog</td>
<td>30 seconds maximum</td>
</tr>
<tr>
<td>Login to SMS main screen</td>
<td>20 seconds maximum</td>
</tr>
<tr>
<td>SMS main screen to ACS</td>
<td>5 seconds maximum</td>
</tr>
<tr>
<td>ACS main screen to CPE</td>
<td>15 seconds maximum</td>
</tr>
</tbody>
</table>

**Screen**

Here is the required screen specification.

<table>
<thead>
<tr>
<th>Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>800 x 600 pixel resolution</td>
</tr>
</tbody>
</table>

**Preparing the System**

**Introduction**

It is recommended that you check the kernel parameters on the system to ensure the system is optimally configured.

The following parameters are described in their respective technical guides. However, they are collated here for reference.

**Note:** Actual kernel parameters may be greater than those listed here.

**Checking Kernel parameters**

Follow these steps to check the Kernel parameters for Solaris.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in as root.</td>
</tr>
<tr>
<td>2</td>
<td>Enter <code>cat /etc/system</code></td>
</tr>
<tr>
<td>3</td>
<td>Check the parameters are set to at least the minimum values.</td>
</tr>
<tr>
<td>4</td>
<td>Change the parameters as required using the following command from <code>/etc/system</code>.</td>
</tr>
</tbody>
</table>

**Parameters**

This table shows all kernel parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min Value</th>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>semmni</td>
<td>1024</td>
<td></td>
<td>Number of semaphore identifiers.</td>
</tr>
</tbody>
</table>
### Parameter Table

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min Value</th>
<th>Hex Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>semmsl</td>
<td>1024</td>
<td></td>
<td>Maximum number of semaphores per unique ID.</td>
</tr>
<tr>
<td>semmns</td>
<td>14000</td>
<td></td>
<td>Maximum number of semaphores.</td>
</tr>
<tr>
<td>shmmax</td>
<td>4294967295</td>
<td>40000000</td>
<td>Maximum shared memory segment (bytes).</td>
</tr>
<tr>
<td>shmmn</td>
<td>1</td>
<td></td>
<td>Minimum shared memory segment (bytes).</td>
</tr>
<tr>
<td>shmmni</td>
<td>400</td>
<td></td>
<td>Number of shared memory identifiers.</td>
</tr>
<tr>
<td>shmseg</td>
<td>50</td>
<td></td>
<td>Number of shared memory segments allowed per process.</td>
</tr>
<tr>
<td>semopm</td>
<td>100</td>
<td></td>
<td>Maximum number of semaphore operations that can be executed per semop system call.</td>
</tr>
<tr>
<td>semvmx</td>
<td>65535</td>
<td></td>
<td>Maximum semaphore value.</td>
</tr>
</tbody>
</table>

### Database Timezone and Backups

#### Setting Oracle timezone

To operate correctly, Oracle must be running on Greenwich mean time (GMT).

To ensure that Oracle is running on GMT, check that the following line is in the Oracle user’s `.profile`:

```
TZ=GMT
export TZ
```

#### Oracle database domain

Check that the sqlnet.ora file does not override the oracle database domain specified in the initSMF*.ora file. Overriding will cause database creation failure with an inability to resolve the required database name in tnsnames.ora.

The initSMF*.ora files are located in the `/IN/service_packages/SMS/db/install/create/SMP/machine-profile` directory.

Each file should contain the following line:

```
db_domain=basmslp.SMF
```

The sqlnet.ora file should contain the following line:

```
NAMES.DEFAULT_DOMAIN = Oracle
```

The sqlnet.ora file will be in the `$ORACLE_HOME/network/admin/` directory.

**Note:** The specific initSMF*.ora file used in the installation is specific during the execution of the smsSms installation script.

#### SMF backups

The SMF can be backed up in two ways.

- Shut down the database periodically and backup all the database data files. This is simple but will disable provisioning and service side updates for the duration of the backup.
- Hot Backups:
  - Archive logging should be enabled
  - Archive logs and table spaces must be backed up and archive logs removed periodically. This procedure must be implemented by an individual with good knowledge of Oracle databases.
Archive logging

It is important to remember that if archive logging is enabled and the archive logs are not removed periodically then the disk will eventually fill up and the database will cease to function.

Starting Oracle Automatically on Reboot

Setting the initialization

In an operational environment, it is desirable that Oracle automatically start on reboot. This requires the creation of various scripts in the Unix initialization directories. A script is provided to simplify this task.

Before you begin

These tasks require that the "dbstart" script is in the default PATH for the "Oracle" user.

Procedure

Follow these steps to configure Oracle on Solaris.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log on as root.</td>
</tr>
<tr>
<td>2</td>
<td><strong>Enter</strong> cd /IN/service_packages/SMS/install/init-sun</td>
</tr>
<tr>
<td>3</td>
<td><strong>Enter</strong> sh ./oracle.sh</td>
</tr>
<tr>
<td></td>
<td><strong>Result:</strong> The script will start.</td>
</tr>
<tr>
<td></td>
<td>The output below shows the steps that are performed to configure automatic starting of Oracle on reboot.</td>
</tr>
<tr>
<td></td>
<td><strong>Action:</strong> Configure Startup/Shutdown</td>
</tr>
<tr>
<td></td>
<td>Removing /etc/init.d/oracle</td>
</tr>
<tr>
<td></td>
<td>Removing /etc/rc.config.d/oracle</td>
</tr>
<tr>
<td></td>
<td>Removing /etc/rc2.d/S901oracle</td>
</tr>
<tr>
<td></td>
<td>Removing /etc/rc0.d/K011oracle</td>
</tr>
<tr>
<td></td>
<td>Copying oracle to /etc/init.d/oracle</td>
</tr>
<tr>
<td></td>
<td>/etc/rc.config.d exists</td>
</tr>
<tr>
<td></td>
<td>Creating /etc/rc.config.d/oracle</td>
</tr>
<tr>
<td></td>
<td>Creating link to /etc/rc2.d/S901oracle</td>
</tr>
<tr>
<td></td>
<td>Creating link to /etc/rc0.d/K011oracle</td>
</tr>
</tbody>
</table>
Chapter 11

About Installation and Removal

Overview

Introduction

This chapter provides details of the installation and removal process for the application.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation and Removal Overview</td>
<td>193</td>
</tr>
<tr>
<td>Raw Devices on Clustered SMS</td>
<td>193</td>
</tr>
<tr>
<td>Setting up ssh keys</td>
<td>196</td>
</tr>
<tr>
<td>Checking the Installation</td>
<td>197</td>
</tr>
</tbody>
</table>

Installation and Removal Overview

Introduction

For information about the following requirements and tasks, see *NCC Installation Guide*:

- NCC system requirements
- Pre-installation tasks
- Installing and removing NCC packages

SMS packages

An installation of Service Management System includes the following packages, on the:

- SMS:
  - smsSms
  - smsCluster (clustered)
  - efmSms
  - efmCluster (clustered)
- SLC:
  - smsScp
- VWS:
  - smsExtras

Raw Devices on Clustered SMS

Raw devices

SMS can allocate tablespace storage based on raw (without a file system) partitions. This enhances the performance of SMS on the SMS.
If you are using the raw devices option, you must create the raw partitions before installing the database using tools such as the system’s format command.

The raw devices file (which you will be prompted to complete during the installation) must contain the full paths of the device files for the appropriate partitions.

The partitions must be at least as big as the required datafile sizings listed in the sizing file which will be used by the installation.

**Example smf_devices.sh file**

This is an example smf_devices.sh file.

```bash
#!/bin/sh
# The following file is the structure required for knowledge of
# raw device utilization and a few details pertaining to cluster
# creation. If clusters are not used then retaining the default
# values will be sufficient and not impact installation for raw
# device only.
#
# Details about the cluster
# How many instances in the cluster?
CLUSTER_INSTANCES=2
export CLUSTER_INSTANCES

# For each instance in the cluster we need to know the node name
# to install into the service configuration
NODE_1=smp1
NODE_2=smp2
export NODE_1 NODE_2

# These are the generic RAW DEVICE requirements for the cluster
# NOTE: ENSURE ALL THESE DEVICES ARE READ WRITEABLE BY THE
# ORACLE USER OTHERWISE INSTALLATION WILL FAIL
# System tablespace
SYSTEM_TABLESPACE=/dev/did/rdsk/d10s5
export SYSTEM_TABLESPACE

# USERS tablespace
USERS_TABLESPACE=/dev/did/rdsk/d10s6
export USERS_TABLESPACE

# Temporary tablespace
TEMP_DATAFILE_1=/dev/did/rdsk/d10s2
TEMP_DATAFILE_2=/dev/did/rdsk/d10s3
TEMP_DATAFILE_3=/dev/did/rdsk/d10s4
TEMP_DATAFILE_4=/dev/did/rdsk/d11s0
TEMP_DATAFILE_5=/dev/did/rdsk/d11s1
TEMP_DATAFILE_6=/dev/did/rdsk/d11s2
TEMP_DATAFILE_7=/dev/did/rdsk/d11s3
TEMP_DATAFILE_8=/dev/did/rdsk/d11s4
export TEMP_DATAFILE_1 TEMP_DATAFILE_2 TEMP_DATAFILE_3
export TEMP_DATAFILE_4 TEMP_DATAFILE_5 TEMP_DATAFILE_6
export TEMP_DATAFILE_7 TEMP_DATAFILE_8

# Tools tablespace
TOOLS_TABLESPACE=/dev/did/rdsk/d10s7
export TOOLS_TABLESPACE

# 3 control file devices
CONTROL_FILE_1=/dev/did/rdsk/d12s3
CONTROL_FILE_2=/dev/did/rdsk/d12s4
CONTROL_FILE_3=/dev/did/rdsk/d12s5
```

export CONTROL_FILE_1 CONTROL_FILE_2 CONTROL_FILE_3

# Service Configuration Device
SRVM=/dev/did/rdsk/d11s5
export SRVM

# Now the UNDO tables. There will be 1 UNDO tablespace per instance in the cluster, having 5 datafiles per tablespace
# Standard to use here is UNDOTBS\{NODEID\}_DATAFILE_X, so UNDOTBS1
# is the UNDO space for NODE_1
# If clusters are not in use and this is raw device then UNDOTBS1
# only needs populating.
UNDOTBS1_DATAFILE_1=/dev/did/rdsk/d9s0
UNDOTBS1_DATAFILE_2=/dev/did/rdsk/d9s1
UNDOTBS1_DATAFILE_3=/dev/did/rdsk/d9s2
UNDOTBS1_DATAFILE_4=/dev/did/rdsk/d9s3
UNDOTBS1_DATAFILE_5=/dev/did/rdsk/d9s4
export UNDOTBS1_DATAFILE_1 UNDOTBS1_DATAFILE_2 UNDOTBS1_DATAFILE_3
export UNDOTBS1_DATAFILE_4 UNDOTBS1_DATAFILE_5

# We require one of the following UNDOTBS sections PER cluster instance
# The **REQUIRED** format is UNDOTBSX_DATAFILE_Y= where X is the instance
# ID of the node defined in NODE_X and Y is the log file number
UNDOTBS2_DATAFILE_1=/dev/did/rdsk/d9s5
UNDOTBS2_DATAFILE_2=/dev/did/rdsk/d9s6
UNDOTBS2_DATAFILE_3=/dev/did/rdsk/d9s7
UNDOTBS2_DATAFILE_4=/dev/did/rdsk/d10s0
UNDOTBS2_DATAFILE_5=/dev/did/rdsk/d10s1
export UNDOTBS2_DATAFILE_1 UNDOTBS2_DATAFILE_2 UNDOTBS2_DATAFILE_3
export UNDOTBS2_DATAFILE_4 UNDOTBS2_DATAFILE_5

# And the redo logs. The sizing is for $REDO_LOGS_PER_NODE redo logs per node in the cluster, so this section requires $CLUSTER_INSTANCES * $REDO_LOGS_PER_NODE to be complete. Naming standard is redo$NODEID_X, for example REDO1_1, REDO1_2 ...
REDO_LOGS_PER_NODE=16
export REDO_LOGS_PER_NODE

REDO1_1=/dev/did/rdsk/d5s0
REDO1_2=/dev/did/rdsk/d5s1
REDO1_3=/dev/did/rdsk/d5s2
REDO1_4=/dev/did/rdsk/d5s3
REDO1_5=/dev/did/rdsk/d5s4
REDO1_6=/dev/did/rdsk/d5s5
REDO1_7=/dev/did/rdsk/d5s6
REDO1_8=/dev/did/rdsk/d5s7
REDO1_9=/dev/did/rdsk/d6s0
REDO1_10=/dev/did/rdsk/d6s1
REDO1_11=/dev/did/rdsk/d6s2
REDO1_12=/dev/did/rdsk/d6s3
REDO1_13=/dev/did/rdsk/d6s4
REDO1_14=/dev/did/rdsk/d6s5
REDO1_15=/dev/did/rdsk/d6s6
REDO1_16=/dev/did/rdsk/d6s7
export REDO1_1 REDO1_2 REDO1_3 REDO1_4
export REDO1_5 REDO1_6 REDO1_7 REDO1_8
export REDO1_9 REDO1_10 REDO1_11 REDO1_12
export REDO1_13 REDO1_14 REDO1_15 REDO1_16

# As with the UNDOTBS we require a set of redo logs per nodal instance in the cluster. The format **REQUIRED** is REDOX_Y= where X is the instance
# ID of the node defined in NODE_X and Y is the log file number
Setting up ssh keys

Introduction

Some of the processes in SMS use ssh and scp to transfer data around the network. Consequently, ssh keys and permissions need to be set up on the relevant machines.

Procedure

Follow these steps to generate an automatic ssh access to a replication node.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log into the SMS host as smf_oper.</td>
</tr>
</tbody>
</table>
| 2    | Enter ssh smf@host  
    where host stands for the replicated node host, for example, XXSCP1 |
## Chapter 11

### About Installation and Removal

#### Step 3
Run the `ssh-keygen` package.

**Example command:** `ssh-keygen`

**Result:** The script will display the following prompts one at a time:

- Enter file in which to save the key (`/IN/service_packages/SMS/.ssh/id_rsa`):
- Generating public/private rsa key pair.
- Enter passphrase (empty for no passphrase):
- Enter same passphrase again:
- Press Enter to continue.

**Result:** The ssh public key will be generated and saved in `.ssh/id_rsa.pub`.

#### Step 4
Log into the replicated node host, for example, XXSCP1, as `smf_oper`.

#### Step 5
Append the content of the public key to authorized keys.

**Example command:** `cat .ssh/id_rsa.pub >> .ssh/authorized_keys`

#### Step 6
Test the ssh access on the replicated node.

**Example command:** `ssh smf_oper@host`

### Checking the Installation

#### Introduction

Refer to these checking procedures to ensure that SMS has installed correctly.

The end of the smsSms installation process (both unclustered and clustered) specifies a script designed to check the installation just performed. They must be run from the command line.

#### Check unclustered SMS procedure

Follow these steps to ensure SMS has been installed on an unclustered SMS machine correctly.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in to SMS machine as root.</td>
</tr>
</tbody>
</table>
| 2    | Check the following directory structure exists with subdirectories:  
  - `/IN/service_packages/SMS`  
  - `/IN/html` |
| 3    | Check both directories contain subdirectories and that all are owned by:  
  smf_oper user (group oracle) |
| 4    | Log into the system as smf_oper. |

**Note:** This step is to check that the smf_oper user is valid.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 5    | Check that the permissions for smf_oper's .ssh directory are:  
  `dwrx------` |

**Note:** These permissions are required for the ssh keys to work correctly.
### NCC Service Management System Technical Guide

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
</table>
| 6    | **Type** `sqlplus /`  
      No password is required. |
|      | **Note:** This step is to check that the smf_oper user has valid access to the database. |
| 7    | Check the smsSms.install.log has finished with the lines  
      * To validate this install, run:  
        * `smsInstallCheck -u smf/smf -f`  
        `/IN/service_packages/SMS/etc/smsSms.check` |
|      | **Note:** This log will contain 375 (at Version 3.0.0.15) spurious Errors related to drop table/view and alter table commands where no object exits. This is caused by underlying Oracle routines that always drop before creation and can be safely ignored. |
| 8    | Ensure that the required SMS tables have been added to the database.  
      To do this, execute the smsInstallCheck program with the following parameters:  
      ~smf_oper/bin/smsInstallCheck -u smf/smf_password -f  
      `/IN/service_packages/SMS/etc/smsSms.check`  
      **Result:** You should see the following output:  
      
      - **CHECKING:** This program is run with `-u SMF/<password>`.  
        ... OK  
      - **CHECKING:** User smf_oper program smsMaster has 2 instances.  
        ... OK  
      - **CHECKING:** User smf_oper program smsReportsDaemon has 1 instances.  
        ... OK  
      - **CHECKING:** User smf_oper program smsReportScheduler has 1 instances.  
        ... OK  
      - **CHECKING:** User smf_oper program smsAlarmRelay has 1 instances.  
        ... OK  
      - **CHECKING:** User smf_oper program smsNamingServer has 1 instances.  
        ... OK  
      - **CHECKING:** User smf_oper program smsAlarmDaemon has 1 instances.  
        ... OK  
      - **CHECKING:** User _oper program smsTaskAgent has 1 instances.  
        ... OK  
      - **CHECKING:** Logged OraUser (SMF) has 23 tables like 'SMF_%'.  
        ... OK  
      - **CHECKING:** Logged OraUser (SMF) has 10 tables like 'REP_%'.  
        ... OK  
      - **CHECKING:** Logged OraUser (SMF) has 1 tables like 'IORS'.  
        ... OK  
      - **CHECKING:** Logged OraUser (SMF) has 62 valid triggers like 'SMF_%'.  
        ... OK  
      - **CHECKING:** Logged OraUser (SMF) has 22 valid triggers like 'REP_%'.  
        ... OK  
      - **CHECKING:** Logged OraUser (SMF) has 2 valid triggers like 'IORS_%'.  
        ... OK  
      - **CHECKING:** Logged OraUser (SMF) has 1 valid triggers like 'SSD_%'.  
        ... OK  
      - **CHECKING:** Logged OraUser (SMF) has 1 valid triggers like 'SSR %'.  
        ... OK  |
### Check clustered SMS procedure

Follow these steps to ensure SMS has been installed on a clustered SMS machine correctly.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in to SMS machine as root.</td>
</tr>
<tr>
<td>2</td>
<td>Check the following directory structure exists with subdirectories:</td>
</tr>
<tr>
<td></td>
<td>* /IN/service_packages/SMS</td>
</tr>
<tr>
<td></td>
<td>* /IN/html</td>
</tr>
<tr>
<td>3</td>
<td>Check both directories contain subdirectories and that all are owned by:</td>
</tr>
<tr>
<td></td>
<td>smf_oper user (group oracle)</td>
</tr>
<tr>
<td>4</td>
<td>Log into the system as smf_oper.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This step is to check that the smf_oper user is valid.</td>
</tr>
<tr>
<td>5</td>
<td>Check that the permissions for smf_oper's .ssh directory are:</td>
</tr>
<tr>
<td></td>
<td>dwrx------</td>
</tr>
<tr>
<td>Step</td>
<td>Action</td>
</tr>
<tr>
<td>------</td>
<td>--------</td>
</tr>
<tr>
<td><strong>Note:</strong> These permissions are required for the ssh keys to work correctly.</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> This step is to check that the smf_oper user has valid access to the database.</td>
<td></td>
</tr>
</tbody>
</table>

6 Type `sqlplus /`  
No password is required.

7 Check the `smsSms.install.log` has finished with the lines  
* To validate this install, run:  
  * `smsInstallCheck -u smf/smf -f /IN/service_packages/SMS/etc/smsSms.check`  

**Note:** This log will contain 375 (at Version 3.0.0.15) spurious Errors related to drop table/view and alter table commands where no object exits. This is caused by underlying Oracle routines that always drop before creation and can be safely ignored.

8 Ensure that the required SMS tables have been added to the database.  
To do this, execute the `smsInstallCheck` program with the following parameters:  
`~smf_oper/bin/smsInstallCheck -u smf_password -f /IN/service_packages/SMS/etc/smsSms.check`  
**Result:** You should see the following output:

CHECKING: This program is run with `-u SMF/password`.  
... OK

CHECKING: User `smf OPER` program `smsMaster` has 2 instances.  
... OK

CHECKING: User `smf OPER` program `smsReportsDaemon` has 1 instances.  
... OK

CHECKING: User `smf OPER` program `smsReportScheduler` has 1 instances.  
... OK

CHECKING: User `smf OPER` program `smsAlarmRelay` has 1 instances.  
... OK

CHECKING: User `smf OPER` program `smsNamingServer` has 1 instances.  
... OK

CHECKING: User `smf OPER` program `smsAlarmDaemon` has 1 instances.  
... OK

CHECKING: Logged OraUser (SMF) has 23 tables like 'SMF_%'.  
... OK

CHECKING: Logged OraUser (SMF) has 10 tables like 'REP_%'.  
... OK

CHECKING: Logged OraUser (SMF) has 1 tables like 'IORS'.  
... OK

CHECKING: Logged OraUser (SMF) has 62 valid triggers like 'SMF_%'.  
... OK

CHECKING: Logged OraUser (SMF) has 22 valid triggers like 'REP_%'.  
... OK

CHECKING: Logged OraUser (SMF) has 2 valid triggers like 'IORS_%'.  
... OK

CHECKING: Logged OraUser (SMF) has 1 valid triggers like 'SSD_%'.  
... OK
Chapter 11

Check SLC procedure

Follow these steps to ensure SMS has been installed on the SLC machine correctly.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log in to SLC machine as root.</td>
</tr>
</tbody>
</table>
| 2    | Check the following directory structure exists with subdirectories: 
|      | /IN/service_packages/SLEE. |
| 3    | Check both directories contain subdirectories and that all are owned by: 
|      | smf_oper user (group oracle) |
| 4    | Log into the system as smf_oper. |

**Note:** This step is to check that the smf_oper user is valid.

| 5    | Check that the permissions for smf_oper's .ssh directory are: 
|      | `dwrx------` |

**Note:** These permissions are required for the ssh keys to work correctly.

CHECKING: Logged OraUser (SMF) has 1 valid triggers like 'SSR_$'.
... OK
Dec 29 21:30:34 cmnError(29188) NOTICE: smsInstallCheck: Passed all 16 tests.

Note: The smsInstallCheck program will report missing processes. This is because the cluster resources have not been set up yet by smsCluster.

9
Check the entries of the `/etc/inittab` file.

Inittab entries reserved for SMS on SMS:

1  sms7  /IN/service_packages/SMS/bin/smsMasterStartup.sh 
   (runs smsMaster)

10 Ensure the following shell scripts are configured to be run by the clustering software:

1  /IN/service_packages/SMS/bin/smsAlarmDaemonSmsCluster.sh 
   (runs smsAlarmDaemon)
2  /IN/service_packages/SMS/bin/smsAlarmRelayCluster.sh 
   (runs smsAlarmRelay)
3  /IN/service_packages/SMS/bin/smsStatsThresholdCluster.sh 
   (runs smsStatsThreshold)
4  /IN/service_packages/SMS/bin/smsReportSchedulerCluster.sh 
   (runs smsReportScheduler)
5  /IN/service_packages/SMS/bin/smsReportsDaemonCluster.sh 
   (runs smsReportsDaemon)
6  /IN/service_packages/SMS/bin/smsNamingServerCluster.sh 
   (runs smsNamingServer)
7  /IN/service_packages/SMS/bin/smsTaskAgentCluster.sh 
   (runs smsTaskAgent)

11 Check that the processes listed in the process lists are running on the relevant machine.
## Chapter 11

### Step 6

Type `sqlplus /

No password is required.

**Note:** This step is to check that the smf_oper user has valid access to the database.

### Step 7

Ensure that the required ACS triggers have been added to the database for the ACS_ADMIN oracle user.

### Step 8

Check the entries of the `/etc/inittab` file.

Inittab entries reserved for SMS on SLC:

1. `scp1 /IN/service_packages/SMS/bin/cmnPushFilesStartup.sh (runs cmnPushFiles)
2. `scp2 /IN/service_packages/SMS/bin/infMasterStartup.sh (runs infMaster)
3. `scp3 /IN/service_packages/SMS/bin/smsStatsDaemonStartup.sh (runs smsStatsDaemon)
4. `scp4 /IN/service_packages/SMS/bin/smsAlarmDaemonScpStartup.sh (runs smsAlarmDaemon)
5. `scp5 /IN/service_packages/SMS/bin/updateLoaderStartup.sh (runs updateLoader)

### Step 9

Check that the processes listed in the process lists are running on the relevant machine.

### Check other machines procedure

Follow these steps to ensure SMS has been installed correctly on machines other than SMSs or SLCs.

### Step 1

Log in to the machine as root.

### Step 2

Check the following directory structure exists with subdirectories:

`/IN/service_packages/SMS`

### Step 3

Check both directories contain subdirectories and that all are owned by:

smf_oper user (group oracle)

### Step 4

Log into the system as smf_oper.

**Note:** This step is to check that the smf_oper user is valid.

### Step 5

Check that the permissions for smf_oper's .ssh directory are:

```
dwrx------
```

**Note:** These permissions are required for the ssh keys to work correctly.

### Step 6

If a database has been installed on the machine, and SMS statistics has been configured to use the database, type `sqlplus /

No password is required.

**Note:** This step is to check that the smf_oper user has valid access to the database.

### Step 7

Ensure that the required SMS tables have been added to the database for the SMF oracle user.

### Step 8

Check the entries of the `/etc/inittab` file:

Inittab entries reserved for SMS on SLC:
Step | Action
--- | ---
1 | `ext8` /IN/service_packages/SMS/bin/smsStatsDaemonStartup.sh (runs smsStatsDaemon)
2 | `ext9` /IN/service_packages/SMS/bin/smsAlarmDaemonStartup.sh (runs smsAlarmDaemon)
9 | Check that the processes listed in the process lists are running on the relevant machine. For a list of the processes which should be running, see *Process list - other machines* (on page 204).

**Process list**

If the application is running correctly, the following processes should be running on each SMS, started from the inittab:

- smsMaster
- smsMergeDaemon
- smsAlarmDaemon
- smsAlarmManager
- smsAlarmRelay
- smsStatsThreshold
- smsReportScheduler
- smsReportsDaemon
- smsNamingServer
- smsTaskAgent
- smsTrigDaemon

**Process list - clustered SMS**

If the application is running correctly, the following processes should be running on each SMS.

- smsMaster, started from the inittab.
- The following are started by the clustering software.
  - smsAlarmDaemon
  - smsAlarmManager
  - smsAlarmRelay
  - smsStatsThreshold
  - smsReportScheduler
  - smsReportsDaemon
  - smsNamingServer
  - smsTaskAgent
  - smsTrigDaemon

**Process list - SLC**

If the application is running correctly, the following processes should be running on each SLC, started from the inittab:

- infMaster
- updateLoader
- smsAlarmDaemon
- smsStatsDaemon
• cmnPushFiles

**Process list - other machines**

If the application is running correctly, the following processes should be running on each platform, started from the initab:

• If alarms use replication, smsAlarmDaemon
• If statistics use replication, smsStatsDaemon

**Check the SMS Administration Screens**

Check that the SMS administration screens are working correctly. Use an internet browser to open the following web page:

`http://smshostname`  
Where:  
`smshostname` is the hostname of an SMS in the IN.

Launch the application using the Webstart link. For more information about using the SMS Java administration screens, see *SMS User’s Guide*.

**Check alarm replication**

Follow these steps to check that alarm replication is functioning correctly.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Open the SMS Java administration screen.</td>
</tr>
<tr>
<td>2</td>
<td>Open the <em>Operator Functions</em> &gt; Node Management screen.</td>
</tr>
</tbody>
</table>
| 3    | Click Create.  
  **Result:** This will create a `replication.config` file and distribute a copy to all the machines in the IN. |
| 4    | Check that the file exists on all the machines in the IN. |
| 5    | On each node in the IN, use smsLogTest to generate an error. For more information about smsLogTest, see *smsLogTest* (on page 160). |
| 6    | Open the *Operator Functions* > Alarm Management screen. |
| 7    | Check that the alarm has replicated from each node into the SMF_ALARM_MESSAGE table in the SMF. |

**Enabling index defragmentation**

Once the installation process is completed, it is advisable to enable the index defragmentation facility, although it is not strictly necessary.

*Note:* This facility has a dependency on specific Oracle configuration settings which relate to the nature of deployment. For more information how to install the defragmentation script, see *Index defragmentation*. 
NCC Glossary of Terms

ACS
Advanced Control Services configuration platform.

ANI
Automatic Number Identification - Term used in the USA by long-distance carriers for CLI.

API
Application Programming Interface

ASN.1
Abstract Syntax Notation One - a formal notation used for describing data transmitted by telecommunications protocols. ASN.1 is a joint ISO/IEC and ITU-T standard.

CCS
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.

CDR
Call Data Record

Note: The industry standard for CDR is EDR (Event Detail Record). Over time EDR will replace CDR in the Oracle documentation.

CLI
Calling Line Identification - the telephone number of the caller. Also referred to as ANI.

Connection
Transport level link between two peers, providing for multiple sessions.

CORBA
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.

CPE
Control Plan Editor (previously Call Plan Editor) - software used to define the logic and data associated with a call - for example, "if the subscriber calls 0800 nnnnnn from a phone at location xxx then put the call through to bb bbb bbbb".
CPU
Central Processing Unit

cron
Unix utility for scheduling tasks.

crontab
File used by cron.

DB
Database

DTMF
Dual Tone Multi-Frequency - system used by touch tone telephones where one high and one low frequency, or tone, is assigned to each touch tone button on the phone.

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 NCC 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.

HTTP
Hypertext Transport Protocol is the standard protocol for the carriage of data around the Internet.

IN
Intelligent Network

initab
The initab holds the start up and configuration information for most of the processes used in SMS. The default location is /etc/initab.
IOR

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, that is, IORs, from the central registry if they want to send a request to a server.

IP

1) Internet Protocol
2) Intelligent Peripheral - This is a node in an Intelligent Network containing a Specialized Resource Function (SRF).

IP address

Internet Protocol Address - network address of a card on a computer

ITU

International Telecommunication Union

MID

Measurement ID - used in Number Portability, counts the occurrences of an error.

Oracle

Oracle Corporation

ORB

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.

PI

Provisioning Interface - used for bulk database updates/configuration instead of GUI based configuration.

PIN

Personal Identification Number

SCP

Service Control Point. Also known as SLC.

SGML

SLC
Service Logic Controller (formerly UAS).

SLEE
Service Logic Execution Environment

SLPI
Service Logic Program Instance

SMP
Service Management Platform (also referred to as SMS).

SMS
Depending on context, can be:
- Short Message Service
- Service Management System platform
- NCC Service Management System application

SN
Service Number

SNMP

SQL
Structured Query Language - a database query language.

SRF
Specialized Resource Function - This is a node on an IN which can connect to both the SSP and the SLC and delivers additional special resources into the call, mostly related to voice data, for example play voice announcements or collect DTMF tones from the user. Can be present on an SSP or an Intelligent Peripheral (IP).

SSF
Sub Service Field.

SSP
Service Switching Point

SUA
Signalling Connection Control Part User Adaptation Layer
**System Administrator**

The person(s) responsible for the overall set-up and maintenance of the IN.

**TCAP**

Transaction Capabilities Application Part – layer in protocol stack, message protocol.

**TCP**

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.

**USMS**

Universal Service Management System hardware platform.

**VWS**

Oracle Voucher and Wallet Server (formerly UBE).

**XML**

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 `metalanguage` — 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.
Index

A
-a <days> • 125
-a dir • 137
About applet parameters in .html files • 70
About customizing the SMS UI • 70
About Installation and Removal • 199
About smsStatQuery • 174
About This Document • vii
Accessing SMS • 70
ACS • 213
address • 159
Alarm replication and buffering • 12, 79
Alarm Table fields • 65
Alarms • 4, 9
Alarms diagram • 10
Alarms replication process • 10
All nodes connected • 26
All nodes isolated • 29
ANI • 213
API • 213
applicationName • 106, 107
Architectural overview • 8
Archive logging • 198
ASN.1 • 213
Audience • vii
Auditing • 47
Auditing - listAudit.sh • 47

B
Background Processes on the SLC • 121
Background Processes on the SMS • 77
Before you begin • 192, 198
Boolean expressions • 170

C
-c • 112
-C <secs> • 127
-c dir • 137
-C n • 135
CCS • 213
CDR • 213
Check alarm replication • 211
Check clustered SMS procedure • 206
Check other machines procedure • 209
Check SLC procedure • 208
Check the SMS Administration Screens • 211
Check unclustered SMS procedure • 203
Checking configuration files • 190
Checking current processes • 189
Checking Kernel parameters • 196
Checking the Installation • 203
CLI • 213
clusterDatabaseHost • 72
cmnConfigRead • 78
cmnConfigSyntaxCheck • 143
cmnPushFiles • 16, 78, 122
cmnReceiveFiles • 16, 78
cmnSU • 144
COMMIT BUSY TIME • 31
COMMIT IDLE TIME • 31
Common Troubleshooting Procedures • 189
community • 60
compareNode • 44, 144
comparisonServer • 145
comparisonServer is failing • 192
Components • 9
CONCAT function • 170
COND function • 171
CONFIG DIR • 31
Configuration • 79, 81, 86, 88, 93, 112, 122,
129, 143, 144, 145, 147, 148, 150, 151, 154,
164, 165, 166, 167, 174, 176, 186
Configuration components • 50
Configuration Overview • 49
Configuration process overview • 49
Configuring Nodes • 74
Configuring Replication Files • 55
Configuring the Environment • 49
Configuring the Oracle Listener • 56
Configuring the Resource Group in the Clustered Environment • 51
Configuring the SNMP Agent • 50, 58, 83
Configuring the snmp.cfg file • 59
CONN RETRY TIME • 32
Connection • 213
CONNECTION TIMEOUT • 32
Control plan execution requests • 118
Copyright • ii
CORBA • 213
CORBA service port • 112
CPE • 213
CPU • 214
cron • 214
crontab • 214

D
-d <dir> • 122, 123, 125
-d rows • 133
Data consistency check • 118
Data flow • 3, 19
Database Auditing • 186
Database comparison components • 44
Database comparison diagram • 44
Database comparison process • 45
Database Comparisons • 43
Database resynchronization components • 46
Database resynchronization diagram • 46
Database resynchronization process • 47
Database Resynchronizations • 45
Database tables • 179
Database Timezone and Backups • 197
databaseHost • 72
databaseID • 71
DB • 214
Default.lang • 67
Default.SMS.hs • 68
Defining the Help Screen Language • 68
Defining the Screen Language • 67
Description • 1, 41, 43, 45, 47, 99, 192
Description of processes and executables • 11, 13, 16
Description of resync processes and executables • 30
Directory structure and filenames • 16
Disabling defragmentation • 192
Document Conventions • viii
DTMF • 214
Dual Network Connection • 6
E
-e • 84, 126
-e secs • 133
EDR • 214
EDR file transfer diagram • 15
EDR file transfer process • 15
EDR format • 88
EDR intermediate file format • 17
EDRs • 15
Enabling defragmentation • 192
Enabling index defragmentation • 211
Enhanced Fault Management • 11
Event name file format • 108
Event statistics • 106
eventEndDateTime • 105
eventName • 104
eventResetBaseTime • 104
Events • 104
eventStartDateDateTime • 105
eventWritePeriod • 106
Example • 127
Example 1 • 168
Example 2 • 168
Example helpset language • 69
Example jnlp resources and applet parameters • 73
Example replication.config • 38
Example replication.def file • 37, 56
Example report script • 183
Example screen language • 67
Example smf_devices.sh file • 200
extraFIFO • 116, 117
F
-F • 125, 134
-f <dir> • 122, 125
-f dir/file • 133
Failover • 25
Failover and Error Recovery • 22
Failure • 78, 81, 82, 85, 86, 88, 90, 92, 93, 94, 95, 98, 99, 111, 112, 114, 117, 122, 127, 128, 130, 131, 138, 142, 145, 149, 150, 164, 166, 176, 187
File Format example 1 • 172
File Format example 2 • 172
File Format example 3 • 173
Format characters • 172
Format File configuration • 169
Formatting as a SNMP trap message • 63
Functions • 1, 169
Further information • 22, 39, 173
G
Generating replication.config • 38
group • 161
Groups • 161
groups-cover-table-on-scp • 160
GUI • 214
H
-h <host> • 123
-h ratio • 134
HB PERIOD • 32
HB TIMEOUT • 32
HB TOLERANCE • 33
Header tags • 169
Heartbeating • 6
HTML • 214
HTML DIR • 33
HTTP • 214
I
-i • 135
-i <dir> • 96
-i <host> • 113
id • 159
Implementing change to replication.def • 55
Implementing changes to the replication.config file • 55
IN • 214
Index Defragmentation • 192, 211
inetCompareServer • 42, 146
Inferior Master nodes • 24
infMaster • 23, 127
infoDisplayer • 147
initab • 214
Input file • 156
Input file example • 162
inputBootstrap • 44, 46, 148
Installation and Removal Overview • 199
Installing a Report Script • 180
Installing Additional Applications • 75
Installing the applications • 75
Interactive reports • 96
Introduction • 6, 9, 12, 15, 19, 22, 31, 38, 49, 55, 56, 58, 65, 67, 68, 93, 179, 180, 182, 186, 189, 190, 196, 199, 202, 203
IOR • 215
IP • 215
IP address • 215
Isolated SLC • 27
Isolated SMS • 28
ITU • 215
J
j2se version • 71
Java applet parameters • 71
Java help screen grayed out • 190
Java Webstart • 73
java_arguments • 71
K
key-columns • 160
L
Language files for multi-byte character sets • 67
LANGUAGEID function • 171
Legacy mode configuration - config file • 139
listenPort • 60, 61, 116
Location • 115, 174
logjob.conf • 50, 91, 131
logo • 72
LONG TIMEOUT • 33
M
-M <secs> • 126
-m size • 134, 135
Maintaining Network Connections • 6
Master Controllers • 21
MASTER PORT • 33
MAX PENDING • 34
MAX_CONTROL_PLANS_DISPLAYED • 73
maxFileOpenTime • 102
maxFileSize • 102
MAXMASTERSNODES • 34
-maxpending • 93
Measurement IDs - standard mode • 138
Measurements • 140
Memory • 195
Merging nodes • 29
Message flows • 8
MIB field mappings - SMF_ALARM.DEFN • 66
MIB field mappings - SMF_ALARM.MESSEGE • 66
MID • 215
my-addr • 60
my-oid • 61
N
Network • 195
Node • 158
NODE ID • 34
Node numbers • 25
nodes • 162
Nodes • 20
notification-oid • 61
O
-o <dir> • 97, 125, 126
opaque • 61, 62
Optional Parameters Table • 175, 176
Oracle • 215
Oracle configuration restriction • 193
Oracle database domain • 197
ORACLE USER • 34
oracleLogin • 115
ORB • 215
Order of replication • 75
other-columns • 160
outDir • 101
outDirBucketSize • 102
outDirExpectedFiles • 101
outDirType • 101
outFileName • 102
Output • 78, 81, 82, 85, 86, 88, 89, 90, 91, 92, 93, 94, 95, 98, 99, 111, 112, 114, 117, 122, 127, 128, 130, 131, 138, 142, 144, 145, 146, 147, 150, 151, 163, 164, 166, 174, 177, 187
Overview • 1, 4, 19, 41, 49, 51, 77, 121, 143, 179, 189, 195, 199
P
-p • 84, 94
-P <dir> • 123
-p <port> • 113, 124
Parameters • 78, 83, 89, 90, 92, 94, 95, 96, 98, 110, 112, 115, 128, 131, 132, 141, 196
Parameters - replication.def • 31
Parameters for legacy mode • 136, 137
Parameters for standard mode • 136, 137
param-oid • 61, 62
perform • 156
PL • 9, 215
PIN • 215
Platform Configuration • 4
POLLING INTERVAL • 35
port • 62
Possible Problems • 190
Pre-installation • 195
Preparing the System • 196
Prerequisites • vii
Primary replication nodes • 21
Procedure • 56, 68, 69, 181, 198, 202
Process Descriptions • 3
Process list • 210
Process list - clustered SMS • 210
Process list - other machines • 209, 210
Process list - SLC • 210
produce-final-reports • 157

Q
QUEUE CRIT THRESH • 35
QUEUE ERR THRESH • 35
QUEUE WARN THRESH • 35

R
-r <pref> • 123
-R <secs> • 126
-r node • 136
ranges • 162
Raw devices • 199
Raw Devices on Clustered SMS • 199
Related documents • vii
REP_PATH • 36
repConfigWrite • 150
Replication • 3, 156
Replication Check • 22, 41
Replication check components • 42
Replication check diagram • 42
Replication check process • 43
Replication Checks • 41
Replication component diagram • 23
Replication components • 23
Replication groups • 21
Replication in an Unclustered Installation • 23
Replication is failing • 191
Replication Overview • 3, 19
Replication process • 20
replication.config File • 38, 39, 50, 150, 164, 191
replication.def File • 31, 50, 92
replicationAllowance • 103
REPORT DIR • 36
Report parameter table • 180
Report Script Worked Example • 182
Report Scripts table • 180
report-after • 157
report-directory • 157
Reporting Functions • 3
report-row-number-limit • 157, 158
Reports • 179
Reports Database Tables • 179
Resend Alarms • 85
resetAllEventStatisticsOnStartup • 104
resetInterval • 105, 108
resetStatisticOnStartup • 107
Resource properties • 70
Response times • 196
Restarting running processes using kill • 189
Restarting the smsAlarmRelay • 64
RESYNC DIR • 36
resyncServer • 45, 46, 151
ROUND function • 171

S
-s • 83
-S • 135
-s <port> • 114
-s <secs> • 111, 124
-S <sufx> • 123
-s Kb • 137
scanInterval • 103
Scope • vii
SCP • 215
scps • 117
Screen • 196
SECONDARY DELAY • 36
Setting Oracle timezone • 197
Setting the initialization • 198
Setting up ssh keys • 202
Setting up the Screens • 70
SGML • 215
Single platform component diagram • 6
Single platform configuration • 5
SLC • 216
SLC Nodes • 74
slcBusyTimeout • 116
SLEE • 216
SLPI • 216
SMF AlarmMessage Format • 65
SMF backups • 197
SMF Listen Messages • 67
SMP • 216
SMS • 216
SMS Client Specifications • 195
SMS component diagram • 2
SMS Java screens will not start • 190
SMS Nodes • 74
SMS packages • 199
SMS subsystems • 2
SMS_PORT • 37
smsAlarmDaemon • 11, 12, 52, 79, 128
smsAlarmManager • 12, 81
smsAlarmRelay • 52, 64, 83
smsApplyConfig.sh • 121
smsCdrProcess.sh • 16, 88
smsCompareResyncClient • 30, 42, 43, 45, 46, 151
smsCompareResyncServer • 30, 42, 43, 154
smsConfigDaemon • 85, 86
smsConfigDaemonScript • 86
smsDbCleanup.sh • 89
smsDumpRepConfig • 39, 150, 164
smsLogCleaner • 90, 130
smsLogTest • 165, 211
smsManualRequester • 166

214 NCC Service Management System Technical Guide
smsMaster • 3, 11, 13, 23, 30, 44, 92, 134
smsMergeDaemon • 24, 91
smsNamingServer • 24, 53, 93
smsProcessCdr • 88, 167
smsRecordStatistic • 174
smsReportCleanupStartup.sh • 98
smsReportScheduler • 53, 96
smsReportsDaemon • 4, 11, 13, 16, 54, 94
smsSendConfig.sh • 111
smsStatisticsWriter • 99
smsStatisticsWriter parameters • 100
smsStatisticsWriter structure • 99
smsStatsDaemon • 13, 99, 132, 175
smsStatsDaemon configuration • 132
smsStatsDaemon parameters • 138
smsStatsQuery • 174
smsStatsThreshold • 12, 13, 54, 110
smsTaskAgent • 4, 24, 54, 112
smsTrigDaemon • 7, 114
SN • 216
SNMP • 216
SNMP relaying - switching on • 59
SNMP v1 • 63
SNMP v3 • 64
snmp.cfg example • 59
snmp.cfg file parameters • 60
Special fields • 169
specific • 61, 62
Specifications • 195
Specifying the Statistics to Query • 175
SQL • 216
SRF • 216
SSF • 216
SSP • 216
Standard fields • 169
Starting and stopping • 64
Starting Oracle Automatically on Reboot • 198
Starting the smsAlarmDaemon failover • 52
Starting the smsAlarmRelay failover • 52
Starting the smsNamingServer failover • 53
Starting the smsReportScheduler failover • 53
Starting the smsReportsDaemon failover • 54
Starting the smsStatsThreshold failover • 54
Starting the smsTaskAgent failover • 54
Starting the sshd failover • 51
Starting the webserver failover • 51
startMerge • 30, 91, 176
Startup • 78, 79, 81, 83, 86, 88, 89, 90, 92, 93, 95, 96, 98, 110, 111, 112, 114, 122, 128, 130, 132, 141, 150
statisticName • 106, 107
Statistics • 4, 12, 106
Statistics collected • 14
Statistics collection diagram • 13
Statistics collection process • 14
Statistics file format • 109
Statistics nodes • 74
Statistics shared memory • 138
Statistics thresholds • 11, 14
statResetBaseTime • 108
statsDaemonRestartDelay • 103
statsDaemonStartupTime • 103
STATSKEY • 37
statWritePeriod • 107
stop-on-limit • 158
Strings • 169
SUA • 216
SUBSTR function • 171
Superior Master nodes • 20
Syntax for the stats_config file • 139
System Administrator • 217
System Overview • 1, 77
T
-t • 84
-T • 127, 135
-t <bits> • 124
-t <host> • 113
-t secs • 137
table • 160, 161
tables • 159
TCAP • 217
TCP • 217
tempDir • 100
The replication.config file • 55
The replication.def file • 55
Tools and Utilities • 143
Transmission of the SNMP trap message • 64
trap • 62
Troubleshooting • 189
Typographical Conventions • viii
TZ • 73
U
-u • 83, 94
-u <usr/pwd> • 97, 111, 113
-u usr/pwd • 136
Unclustered platform configuration • 4, 5
Unix utilities • 98
Update Loader nodes • 20
Update queuing • 22
Update Requester nodes • 21, 24
Update requests to primary nodes • 21
updateLoader • 3, 24, 30, 134, 141
updateLoader failure • 22
Updates • 24
Updating smsStatsDaemon measurements • 141
Usage examples • 80, 130
useFIFO • 116, 117
useORB • 115, 116
userName • 63
User-specific language settings • 67
use-SNMPv3 • 60, 61, 63
USMS • 217

V
-\( v \) • 97, 133
variable bindings • 64
view • 158
VWS • 217

W
-\( w \) • 136
-\( w <secs> \) • 114, 124
What is Replication? • 19, 141
What is the Service Management System? • 1

X
-\( x \) • 84, 124
XML • 217
xmlTcapInterface • 9

Z
-\( z <timezone> \) • 97