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 .................................................................................................................. vi
Document Conventions ............................................................................................................. vi

### Chapter 1
**System Overview** .................................................................................................................. 1
- Overview ................................................................................................................................. 1
- Introduction to SLEE TCAP Interfaces ..................................................................................... 1
- SCCP Level TCAP Interfaces .................................................................................................. 2
- M3UA Level TCAP Interfaces .................................................................................................. 3
- Routing to Services .................................................................................................................. 5

### Chapter 2
**Configuration** .......................................................................................................................... 9
- Overview ................................................................................................................................. 9
- Configuration Overview ......................................................................................................... 9
- Configuring tcapif.def .......................................................................................................... 12
- Configuring sigtran.config ................................................................................................... 19
- Example Configuration Scenarios .......................................................................................... 46

### Chapter 3
**Background Processes** .......................................................................................................... 51
- Overview ............................................................................................................................... 51
- sua_if ..................................................................................................................................... 51
- m3ua_if ................................................................................................................................. 52

### Chapter 4
**Troubleshooting** .................................................................................................................... 53
- Overview ............................................................................................................................... 53
- Common Troubleshooting Procedures .................................................................................. 53
- Debug ..................................................................................................................................... 53

### Chapter 5
**Statistics and Reports** .......................................................................................................... 55
- Overview ............................................................................................................................... 55
- Statistics ............................................................................................................................... 55

### Chapter 6
**About Installation and Removal** .......................................................................................... 57
- Overview ............................................................................................................................... 57
- Installation and Removal Overview ....................................................................................... 57
- Checking the Installation ....................................................................................................... 57
NCC Glossary of Terms................................................................. 59
Index ...................................................................................... 65
About This Document

Scope

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

Audience

This guide was written primarily for system administrators and persons installing and administering the SIGTRAN Interfaces 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:

- SLEE Technical Guide
- SMS Technical Guide
### 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. Names of database tables and fields.</td>
</tr>
<tr>
<td><strong>Italics</strong></td>
<td>Name of a document, chapter, topic or other publication. Emphasis within text.</td>
</tr>
<tr>
<td><strong>Button</strong></td>
<td>The name of a button to click or a key to press. Example: To close the window, either click <strong>Close</strong>, or press <strong>Esc</strong>.</td>
</tr>
<tr>
<td><strong>Key+Key</strong></td>
<td>Key combinations for which the user must press and hold down one key and then press another. Example: <strong>Ctrl+P</strong>, or <strong>Alt+F4</strong>.</td>
</tr>
<tr>
<td><strong>Monospace</strong></td>
<td>Examples of code or standard output.</td>
</tr>
<tr>
<td><strong>Monospace Bold</strong></td>
<td>Text that you must enter.</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td>Used to indicate variables or text that should be replaced.</td>
</tr>
<tr>
<td><strong>menu option &gt; menu option &gt;</strong></td>
<td>Used to indicate the cascading menu option to be selected, or the location path of a file. Example: <strong>Operator Functions &gt; Report Functions</strong>. Example: <strong>/IN/html/SMS/Helptext/</strong></td>
</tr>
<tr>
<td><strong>hypertext link</strong></td>
<td>Used to indicate a hypertext link on an HTML page.</td>
</tr>
</tbody>
</table>

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.

- Introduction to SLEE TCAP Interfaces
- SCCP Level TCAP Interfaces
- M3UA Level TCAP Interfaces
- Routing to Services

Introduction to SLEE TCAP Interfaces

Introduction

The Oracle SIGTRAN TCAP Interface is a SLEE interface that enables the SLEE to inter-work with a TCAP Protocol stack.

The interface converts messages arriving from the TCAP Protocol stack and converts them into SLEE events. The SLEE events are then sent to the application which is configured to handle the call. The SIGTRAN TCAP Interface also converts SLEE events coming from a SLEE application back into a form the TCAP Protocol stack can understand.

For more information about SLEE events and applications, see SLEE Technical Guide.

Service routing and message correlation

The SIGTRAN TCAP Interface also has a role in routing calls to services on the platform.

It routes the message setting one of the following:

- The SLEE service key for the message
- A correlation ID which matches the message to one sent to the SLEE earlier (in this case, the second message will use the same service key as the first)

TCAP Protocol stack

A TCAP Protocol stack is a software implementation of a networking protocol suite. It involves a group of protocols working together to allow Oracle platform software and hardware to communicate with a telecommunications network.

The protocols are:
TCAP Interface layers

The SIGTRAN TCAP Interface used in a specific installation will depend on the requirements of the network and the type of physical interface and network protocols used.

Generally, the SIGTRAN TCAP Interface which will be installed is built from underlying layers of smaller protocol stacks which sit below the TCAP layer. These layers may be comprised of TCP/IP or SIGTRAN protocols (SUA/SCCP/M3UA). They may be provided and supported by Oracle and/or third party vendors.

Available TCAP Interfaces

This table lists the available SIGTRAN TCAP Interfaces.

<table>
<thead>
<tr>
<th>Stack name</th>
<th>Protocol</th>
<th>Interface name</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIGTRAN</td>
<td>SCCP</td>
<td>sua_if</td>
</tr>
<tr>
<td></td>
<td>M3UA</td>
<td>m3ua_if</td>
</tr>
</tbody>
</table>

SCCP Level TCAP Interfaces

Introduction

The SIGTRAN TCAP Interface stack can be used to translate to a third party SCCP/SUA implementation.

SIGTRAN SCCP/SUA interface diagram

This diagram shows the SIGTRAN SCCP/SUA interface on a SLC.
**SIGTRAN/SUA interface components**

This table describes the main components in the SUA version of the SIGTRAN TCAP Interface.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>sua_if</td>
<td>Interface between the SLEE and the network.</td>
<td><code>sua_if</code> (on page 51).</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This process includes the SIGTRAN libraries which interface to the network.</td>
<td></td>
</tr>
<tr>
<td>SLEE</td>
<td>Real-time interface between the interfaces and applications which have been configured to communicate through the SLEE.</td>
<td><code>SLEE Technical Guide</code>.</td>
</tr>
<tr>
<td>sua_if.sh</td>
<td>Shell Startup script used to set the command line parameters for configuring sua_if.</td>
<td><code>Configuration Overview</code> (on page 9).</td>
</tr>
<tr>
<td>tcapif.def</td>
<td>Optional configuration file for sua_if. Can be used to set some command line parameters.</td>
<td><code>Configuring tcapif.def</code> (on page 12).</td>
</tr>
<tr>
<td>sigtran.config</td>
<td>Main configuration file for sua_if.</td>
<td><code>Configuring sigtran.config</code> (on page 19).</td>
</tr>
<tr>
<td>sccp_YYYYMMDD_hhmm.log</td>
<td>The SCCP-level message log file.</td>
<td><code>log</code> (on page 25).</td>
</tr>
</tbody>
</table>

**M3UA Level TCAP Interfaces**

**Introduction**

The SIGTRAN TCAP stack can be used to translate to a third party SCCP/M3UA implementation.
SIGTRAN m3ua interface diagram

This diagram shows the SIGTRAN M3UA interface on a SLC.

SIGTRAN m3ua interface components

This table describes the main components in the M3UA version of the SIGTRAN TCAP Interface.

<table>
<thead>
<tr>
<th>Process</th>
<th>Role</th>
<th>Further information</th>
</tr>
</thead>
<tbody>
<tr>
<td>m3ua_if</td>
<td>Interface between the SLEE and the network.</td>
<td>m3ua_if (on page 52).</td>
</tr>
<tr>
<td></td>
<td>Note: This process includes the SIGTRAN libraries which interface to the network.</td>
<td></td>
</tr>
<tr>
<td>SLEE</td>
<td>Real-time interface between the interfaces and applications which have been configured to communicate through the SLEE.</td>
<td>SLEE Technical Guide.</td>
</tr>
<tr>
<td>m3ua_if.sh</td>
<td>Shell Startup script used to set the command line parameters for configuring m3ua_if.</td>
<td>Configuration Overview (on page 9).</td>
</tr>
<tr>
<td>tcapif.def</td>
<td>Optional configuration file for sua_if. Can be used to set some command line parameters.</td>
<td>Configuring tcapif.def (on page 12).</td>
</tr>
<tr>
<td>sigtran.config</td>
<td>Main configuration file for m3ua_if.</td>
<td>Configuring sigtran.config (on page 19).</td>
</tr>
<tr>
<td>sccp_YYYYMMDD_hhmm.log</td>
<td>The SCCP-level message log file.</td>
<td>log (on page 25).</td>
</tr>
</tbody>
</table>
Routing to Services

Introduction

When the SIGTRAN TCAP Interface receives a new TCAP message (TC-BEGIN), it determines what SLEE service key it should use when sending the message on. SLEE service keys are used by the SLEE to determine where to route the message to. For more information about how SLEE routes calls, see SLEE Technical Guide.

**Note:** If the message is an assistRequestInstructions, sua_if/m3ua_if will send the message to the SLEE with a correlation ID. The SLEE will then route based on only correlation ID. For more information about correlation IDs and how they are processed, see Correlation IDs (on page 7).

Routing process

This table describes how SIGTRAN TCAP Interface constructs the SLEE service key for an incoming message.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TC-BEGIN arrives at sua_if/m3ua_if.</td>
</tr>
</tbody>
</table>
| 2     | sua_if/m3ua_if determines which protocol the message is using by matching the SSN the message arrived from to the ssn details in its command line or tcapif.def configuration.  
For more information about setting which SSNs correspond to which protocols, see tcapif parameters (on page 12). |
| 3     | If the protocol is INAP, sua_if/m3ua_if will check whether the operation is assistRequestInstructions. If it is, it will set a correlation ID in the message and send it to the SLEE. No further action is taken. |
| 4     | If the protocol is not INAP, or it is INAP but the operation is an InitialDP, sua_if/m3ua_if will construct the SLEE service key. |
| 5     | sua_if/m3ua_if sends the message on to the SLEE, where it will be routed according to the rules defined for service keys in SLEE.cfg. |
SLEE service key construction

The SLEE service key constructed by sua_if/m3ua_if is made up from the following elements:

<table>
<thead>
<tr>
<th>Byte</th>
<th>MSB 8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>LSB 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sourced from</td>
<td>Base service key value defined by sleekey (on page 18).</td>
<td>Dest SSN (SCCP)</td>
<td>Depends on the protocol of incoming message:</td>
<td>Protocol</td>
<td>Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INAP IDP</td>
<td>IDP's ServiceKey value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>INAP Initiate CallAttempt</td>
<td>ffffffe</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>MAP</td>
<td>MAP Operation ID value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CAMEL GPRS</td>
<td>CAP_InitialDPGPRS Servicekey value, or DestinationReference for other operations</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CAMEL SMS</td>
<td>CAP_InitialDPSMS Servicekey value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>any other</td>
<td>ffffffe</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Example 1: 0x1 0xd0 00000009
Example 2: 0x123456 0x05 ffffffe

Note: The base service key (bytes 6-8) is not padded with leading zeros. Bytes 1 to 4, and byte 5 are padded with leading zeros.

Example SLEE service keys

Example 1:
If sua_if/m3ua_if is using the default base key of 0x1, and the TC-BEGIN has INAP SSN = 13 (that is, 0xd) and service key = 8: the SLEE service key will be 0x10d00000008.
The message can then be routed to INAPService1 on App1 by the following lines in SLEE.cfg:
SERVICEKEY=INTEGER 0x10d00000008 INAPService1
SERVICE=INAPService1 1 App1 INAPService1

Example 2:
If sua_if/m3ua_if is using a non-default base key of 0x1234, and the TC-BEGIN has INAP SSN = 112 (that is, 0x70) and service key = 10: the SLEE service key will be 0x12347000000010.
The message can then be routed to INAPService2 on App2 by the following lines in SLEE.cfg:
SERVICEKEY=INTEGER 0x12347000000010 INAPService2
SERVICE=INAPService2 1 App2 INAPService2

Example 3:
If sua_if/m3ua_if is using the default base key of 0x1, and the TC-BEGIN has MAP SSN = 5 and operation ID = 5: the SLEE service key will be 0x10500000005.
The message can then be routed to MAPService on App2 by the following lines in SLEE.cfg:
SERVICEKEY=INTEGER 0x10500000005 MAPService
SERVICE=MAPService 1 App2 MAPService
Correlation IDs

In some circumstances, a message arriving at sua_if/m3ua_if will need to be matched to an earlier message. For example, when a play announcement node has requested an Intelligent Peripheral to play a message to a caller, and the IP is reporting the result of the action.

In this case, the second message received by sua_if/m3ua_if (sent by the IP) will be an INAP AssistRequestInstructions (ARI) operation, and will contain a correlation ID. sua_if/m3ua_if will attempt to initiate a SLEE dialog using the correlation ID instead of a service key. The correlation ID will be a decimal conversion of the digits from the ARI’s correlationID parameter.

SLEE Correlation ID diagram

This diagram shows how correlation IDs are linked across the system.

Matching SLEE correlation IDs

These are the steps involved in matching correlation IDs.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>On an SLC, an InitialDP is sent across the SLEE. (Usually sent by an interface such as sua_if/m3ua_if to a service application such as slee_acs). The SLEE assigns it a call ID.</td>
</tr>
<tr>
<td>2</td>
<td>The SLEE service application which received the InitialDP requests the SLEE to assign a correlation ID to the InitialDP. The SLEE allocates a correlation ID and returns it to the SLEE service application.</td>
</tr>
<tr>
<td>3</td>
<td>The SLEE service application sends an INAP ETC operation over the SLEE to an interface. The operation contains the Address of an external network entity and the correlation ID.</td>
</tr>
<tr>
<td>4</td>
<td>The interface sends the operation to the switch which forwards it to the remote entity specified in the Address (usually an Intelligent Peripheral).</td>
</tr>
<tr>
<td>5</td>
<td>The remote entity sends the interface on the SLC an INAP TC-BEGIN containing an AssistRequestInstructions (ARI) and the original correlation ID.</td>
</tr>
<tr>
<td>6</td>
<td>When sua_if/m3ua_if receives the ARI, it initiates a new SLEE dialog using the correlation ID. The correlation ID is matched to the call ID assigned in Stage 1, and the</td>
</tr>
<tr>
<td>Stage</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>same service key routing rules are applied (this means the ARI will be delivered to the SLEE service application which sent the ETC). The correlation ID must be read from within the TCAP component which holds the conveyed protocol's first message.</td>
</tr>
<tr>
<td>7</td>
<td>The SLEE service application receives the message with the call ID assigned to the InitialDP and correlates the messages.</td>
</tr>
</tbody>
</table>

For an example of how correlation IDs are used by ACS when playing announcements, see ACS Technical Guide.
Overview

Introduction

This chapter explains how to configure the Oracle Communications Network Charging and Control (NCC) application.

In this chapter

This chapter contains the following topics.

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration Overview</td>
<td>9</td>
</tr>
<tr>
<td>Configuring tcapif.def</td>
<td>12</td>
</tr>
<tr>
<td>Configuring sigtran.config</td>
<td>19</td>
</tr>
<tr>
<td>Example Configuration Scenarios</td>
<td>46</td>
</tr>
</tbody>
</table>

Configuration Overview

Introduction

This topic provides a high level overview of how the SIGTRAN TCAP Interfaces are configured. sua_if/m3ua_if are configured using the following options set in the:

- Startup shell script and/or tcapif.def
- sigtran.config file

Configuration components

TCAP Interface 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>Environmental</td>
<td>any machine connected to</td>
<td>Set in startup scripts.</td>
<td>Environmental variables (on page 10).</td>
</tr>
<tr>
<td>variables</td>
<td>a switch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Command line variables</td>
<td>any machine connected to</td>
<td>Set in startup scripts or tcapif.def.</td>
<td>Configuring tcapif.def (on page 12).</td>
</tr>
<tr>
<td>tcapif.def</td>
<td>any machine connected to</td>
<td>Optional file that sets the configuration parameters which are shared with</td>
<td>Configuring tcapif.def (on page 12).</td>
</tr>
<tr>
<td></td>
<td>a switch</td>
<td>other types of TCAP Interface.</td>
<td></td>
</tr>
<tr>
<td>sigtran.config</td>
<td>any machine connected to</td>
<td>This file sets the configuration parameters which are specific to SIGTRAN</td>
<td>Configuring sigtran.config (on page 19).</td>
</tr>
<tr>
<td></td>
<td>a switch</td>
<td>TCAP Interface.</td>
<td></td>
</tr>
</tbody>
</table>
### Configuration process overview

This table describes the steps involved in configuring the SIGTRAN TCAP Interface for the first time.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1     | The environment sua_if/m3ua_if will run in must be configured correctly. This includes:  
- Setting the location of the SLEE base directory (if this is different from default)  
- Configuring the name of the main configuration file  
- Configuring the location of the SCCP log file  
For more information about configuring environmental variables, see Environmental variables. |
| 2     | Changing the default startup variables in the SLEE.cfg file, especially service keys. |
| 3     | Any non-default command line parameters must be added to one of the following:  
  - The relevant sua_if/m3ua_if startup shell script(s):  
    - sua_if.sh  
    - m3ua_if.sh  
    - (if supported) tcapif.def |
| 4     | Configuration data must be configured for sua_if/m3ua_if in sigtran.config. |

### Environmental variables

The following UNIX shell environment variables can be set.

**SCCP_LOG_FILE**

**Syntax:** SCCP_LOG_FILE=filename  
**Description:** File name prefix, including path, of the optional SCCP log file. A timestamp and .log suffix is added.  
**Type:** String  
**Optionality:** Optional, default used if not set.  
**Allowed:** /tmp/sccp  
**Default:** /tmp/sccp  
**Notes:** Using the default value for example, the resulting file name in the output of log files have names in the format /tmp/sccp_yyyyymmdd_hhmm.log.  
By default a new log file, with a different timestamp, is created every 10 minutes. Other intervals can be specified using the SCCP_LOG_TIME environment variable.  
**Example:** SCCP_LOG_FILE=/var/tmp/sua_if_sccp

**SCCP_LOG_TIME**

**Syntax:** SCCP_LOG_TIME = value  
**Description:** The interval, in minutes, between starting an SCCP_LOG_FILE log file.  
**Type:** Integer  
**Optionality:** Optional (default used if not set).  
**Allowed:** 10 (minutes)  
**Default:** 10 (minutes)  
**Notes:** If SCCP_LOG_TIME is defined as 20 then, if logging is enabled, new log files will be created every 20 minutes on the hour, 20 past and 40 past the hour with names like: /var/tmp/sccp_20091225_1500.log
Example: \texttt{SCCP\_LOG\_TIME = 20}

\textbf{SLEE\_ETC\_DIR}

\textbf{Syntax:} \texttt{SLEE\_ETC\_DIR= path}

\textbf{Description:} Full path name of the SLEE’s etc directory.

\textbf{Type:} Integer

\textbf{Optionality:} Optional, default used if not set.

\textbf{Allowed:} .

\textbf{Default:} .../etc

\textbf{Notes:} SLEE’s etc directory is where tcapif.def resides.

\textbf{Example:} \texttt{SLEE\_ETC\_DIR=../../../SLEE/etc}

\textbf{TCAPIF\_DEF}

\textbf{Syntax:} \texttt{TCAPIF\_DEF= \textit{filename}}

\textbf{Description:} Defines the name of configuration file which contains TCAP Interface parameters common to all TCAP Interfaces.

\textbf{Type:} String

\textbf{Optionality:} Optional, default used if not set.

\textbf{Allowed:} tcapif.def

\textbf{Default:} tcapif.def

\textbf{Notes:}

\textbf{Example:} \texttt{TCAPIF\_DEF=sua_tcapif.def}

\textbf{ESERV\_CONFIG\_FILE}

\textbf{Syntax:} \texttt{ESERV\_CONFIG\_FILE= \textit{filename}}

\textbf{Description:} Name of configuration file which contains SIGTRAN TCAP Interface-specific parameters.

\textbf{Type:} String

\textbf{Optionality:} Optional, default used if not set.

\textbf{Allowed:} Any valid filename.

\textbf{Default:} sigtran.config

\textbf{Notes:}

\textbf{Example:} \texttt{ESERV\_CONFIG\_FILE=sigtran.config}

\textbf{SIGTRAN\_CONFIG\_SECTION}

\textbf{Syntax:} \texttt{SIGTRAN\_CONFIG\_SECTION= \textit{str}}

\textbf{Description:} The name of the section in the ESERV\_CONFIG\_FILE configuration file which holds the Sigtran Tcap Interface-specific configuration.

\textbf{Type:} String

\textbf{Optionality:} Optional (default used if not set).

\textbf{Allowed:} ASCII text.

\textbf{Default:} For m3ua_if: M3UA.

\textbf{Notes:}

\textbf{Example:} \texttt{SIGTRAN\_CONFIG\_SECTION=M3UA}
Configuring tcapif.def

Setting parameters

Each parameter has a default that will be used if the parameter is not defined. Parameters may be defined in two ways:

1. The command line
2. In the tcapif.def file

Note: If a parameter is set in both the command line and the tcapif.def file, the command line setting will be used.

Defining the parameters

For a parameter 'param val', the value may be set in the configuration file with a line such as:

```
PARAM VAL=value
```

Note: Spaces can be inserted into the parameter name in tcapif.def without effect.

Or the parameter may be defined on the command line, for example:

- `sua_if -paramval value`
- `m3ua_if -paramval value`

Note: Spaces can not be inserted into the parameter names on the command line, and command line parameter names are case sensitive.

Setting command line parameters

Since the executable is started by the SLEE, the only way to set the command line parameters is via an intermediate startup shell script. This shell script is pointed to in SLEE.cfg.

**Example 1:** Startup script:

```
#!/bin/sh
exec sua_if -ssns 123 -proto inap -tcap ccitt
```

**Example 2:** Startup script:

```
#!/bin/sh
exec sua_if -pc 160 -ssns 123 -proto inap -tcap \ ccitt
```

Note: To split over a line (as shown in Example 2), use "\" at the end of the line.

**tcapif.def**

The tcapif.def file can be used to define common configuration for all TCAP interfaces running on the same system.

Where different TCAP Interfaces require different configuration, the file-set configuration options can be overridden on the command line on a binary by binary basis. Command line options are set in the startup shell scripts. Every option in the tcap.def file can be overridden in this way.

Note: In the file, the options are all uppercase. On the command line, they are lowercase.

**tcapif parameters**

The generic configuration variables applicable to all TCAP interfaces are listed below:
Note: If any of the following parameters are set (all default to not set) then the TCAP Interface will use a default SCCP Origination Address in the first TC-CONTINUE that it sends out. By default the TCAP Interface will use the SCCP destination address of the first incoming message. This capability can be used to ensure any subsequent messages sent by the far end in the same dialog will be routed to this address. This can be useful when initial messages are sent to aliased addresses and round-robin routed by an STP to a series of SLCs.

alwayssendaddr
Syntax: -alwayssendaddr true|false
Description: Which messages to send the SCCP destination address in.
Type: Boolean
Optionality: Allowed:
true Send the SCCP destination address in all messages.
false Send the SCCP destination address in only the initial message received in a dialog.
Default: false
Notes: If this parameter and sendorigaddr are both set to true, the TCAP Interface will send the destination and origination addresses.
Example: -alwayssendaddr true

autoac
Syntax: -autoac yes|no
Description: Whether to use the Application context of a CCITT white book TCAP message received in the response to the message if the SLEE application does not supply one.
In addition, dialogs initiated by the interface will use the default application context defined by the defoutac variable below.
Type: Boolean
Optionality: Allowed:
yes Use the Application context from the TCAP message if none is supplied when a SLEE application responds to a message.
no Do not attempt to supply an Application context if none is provided.
Default: yes
Notes: Setting to yes, effectively acknowledges the white book application context used.
If defoutac is set, sua_if/m3ua_if will add the default application context if one is not set by the SLEE application or in the TCAP message.
Example: -autoac no

defoutac
Syntax: -defoutac value
Description: The default application context to use if no application context is supplied by the SLEE application or set by autoac.
Type: String
Optionality:
Allowed: none
Missing application contexts will not be set to a default.

oid1,oid2...
The default CCITT white book TCAP application context to use for dialogs initiated by sua_if/m3ua_if.

Default: none

Notes:

Example: -defoutac 1,2,3

displaymonitors
Syntax: -displaymonitors true|false
Description: Whether to log CAPS.
Type: Boolean
Optionality: Allowed: true
New call attempt rates (CAPS) will be logged to stdout at period defined by -reportperiod.
false
Default: false

Notes:

Example: -displaymonitors true
dpause
Syntax: -dpause seconds
Description: Number of seconds to sleep at startup.
Type: Integer
Optionality: Allowed:
Default: 0
Notes: Allows a global session to be attached to the process.
Example: -dpause 7

inapssns
Syntax: -inapssns value1,value2,...
Description: A comma-separated list of the SCCP subsystem numbers (SSNs) that sua_if/m3ua_if will treat as INAP (regardless of the default protocol defined by -proto).
Type: Integer
Optionality: Mandatory
Allowed: 0-255
Default: Empty set
Notes:
Example: -inapssns 33,45,99

mapssns
Syntax: -mapssns value1,value2,...
Description: A comma-separated list of the SCCP subsystem numbers (SSNs) that sua_if/m3ua_if will treat as MAP (regardless of the default protocol defined by -
proto.
Type: Integer
Optionality: Mandatory
Allowed: 0-255
Default: Empty set
Notes:
Example: -mapssns 22,26

monitorperiod
Syntax: -monitorperiod milliseconds
Description: Period call rate rejection monitoring occurs in.
Type: Integer (number of milli-seconds)
Optionality:
Allowed:
Default: 1000
Notes: Default of 1 sec allows -rejectlevel to represent CAPS.
Example: -monitorperiod 1000

polltime
Syntax: -polltime microseconds
Description: Interval between polling interface.
Type: Integer (micro seconds)
Optionality:
Allowed:
Default: 1000
Notes:
Example: -polltime 500

proto
Syntax: -proto protocol
Description: sua_if/m3ua_if will assume messages from SSNs other than those specified in the mapssns, inapssns and ssns parameters will be using this protocol.
Type: String
Optionality:
Allowed:
inap INAP interface
map MAP interface
is41d IS41D interface
Default:
inap
Notes: This parameter affects service key and correlation ID handling. For more information, see Routing to Services (on page 5).
Example: -proto map

rejectlevel
Syntax: -rejectlevel max number of calls
Description: If more than 0, this sets the maximum number of new call attempts that will be
processed within a given interval (as determined by \texttt{-monitorperiod}).

\begin{itemize}
\item Type: \texttt{Integer}
\item Optionality: \texttt{Allowed: 0}
\item Default: \texttt{0}
\item Notes: In conjunction with \texttt{-monitorperiod} it provides a call limiter for the interface.
\item Example: \texttt{-rejectlevel 4000}
\end{itemize}

\texttt{reportperiod}

\begin{itemize}
\item Syntax: \texttt{-reportperiod reject logging period}
\item Description: How often:
\begin{itemize}
\item rejection indications are logged to the alarm system (if rejections are occurring), and
\item call rates are logged (if \texttt{-displaymonitors} is set).
\end{itemize}
\item Type: \texttt{Integer (seconds)}
\item Optionality: \texttt{Allowed:}
\item Default: \texttt{30}
\item Notes: 
\item Example: \texttt{-reportperiod 25}
\end{itemize}

\texttt{retgt}

\begin{itemize}
\item Syntax: \texttt{-retgt "str"}
\item Description: If set to something other than \texttt{none}, a default SCCP Origination Address will be used which will contain this Global Title.
\item Type: \texttt{String}
\item Optionality: \texttt{Allowed:}
\item Default: \texttt{none}
\item Notes: The variables used in the gt are:
\begin{itemize}
\item \texttt{<noa>} Nature of Address
\item \texttt{<address_digits>} Destination/Called Party Number
\item \texttt{<trans_type>} Transmission Type
\item \texttt{<num_plan>} Number Plan
\end{itemize}
\item Example: \texttt{-retgt "1,1,123456789"}
\end{itemize}

\texttt{retni}

\begin{itemize}
\item Syntax: \texttt{-retni = 0|1}
\item Description: Sets the National Indicator in a return address (an ANSI SCCP address rather than an ITU one), created using the retssn/ retpc/ retgt config parameters.
\item Type: \texttt{Integer}
\item Optionality: 
\end{itemize}
Chapter 2

Configuration

Allowed: 0 Set the NI to 0 (ITU).
1 Set the NI to 1 (ANSI).

Default:

Notes: The National Indicator is the first bit of the return address (that is, the SCCP address).

Example: -retni = 1

retpc

Syntax: -retpc point code value
Description: If set to a non-zero value, a default SCCP Origination Address will be used which will contain this Point Code.
Type: Hex or decimal integer

Optionality: Allowed:

Default: 0

Notes:

Example: -retpc 55

retri

Syntax: -retri Routing Indicator
Description: Default SCCP Origination Address’s routing indicator.
Type: Integer

Optionality: Allowed:

Default: 0

Notes: Used in conjunction with retssn, retpc and retgt options.

Example: -retri 1

retssn

Syntax: -retssn SSN
Description: If set to a non-zero value, a default SCCP Origination Address will be used in the first outgoing TC-CONTINUE message which will contain this SSN.
Type: Integer

Optionality: Allowed:

Default: 0

Notes:

Example: -retssn 20

sendorigaddr

Syntax: -sendorigaddr true|false
Description: Whether to send the SCCP origination address in addition to the destination address.
Type: Boolean
Optionality: Allowed: true Send the SCCP origination address. false Do not send the SCCP origination address. Default: false Notes: If statsif is defined, sendorigaddr is set to true regardless of the configuration value set here. Example: -sendorigaddr true

sleekey
Syntax: -sleekey SLEE_Service_Key
Description: Base SLEE service key value to use in base key (bytes 6-8) part of the SLEE service key created by sua_if/m3ua_if to enable the SLEE to route the message to the correct service.
Type: Integer
Optionality: Optional, default will be used if not set.
Allowed: Hex or decimal integer
Default: 0x1
Notes: for more information about how sleekey is used to construct the base key in the SLEE service key, see Routing process (on page 5).
Example: -sleekey 0x89abcd
This example setting will mean the base key in all SLEE service keys created by this sua_if/m3ua_if will be 0x89abcd.

ssns
Syntax: -ssns ssn1,ssn2,...
Description: A comma separated list of SCCP subsystem numbers (SSNs) that the TCAP Interface will register to.
Type: Decimal integer
Optionality: Allowed: 0 to 255
Default: 19
Notes: Example: -ssns 12,33

statsif
Syntax: -statsif none|slee_if_name
Description: SLEE interface to which all initial messages in a TCAP dialog are copied to allow statistics monitoring.
Type: String
Optionality: Allowed: none don't copy initial messages
SLEE interface to copy initial messages to.
Default: none
Notes: Currently this is only used with the SLEE Callmapping solution. If statsif is not none, sendorigaddr's value is over-ridden and set to true. For more information about SLEE interfaces including the statistics interface, see
Example:
- `statsif none`

**stderron**

Syntax: `-stderron 0|1`

**Description:** Whether syslog messages generated by the interface class should be printed to stderr as well as the system log file.

**Type:** Boolean

**Optionality:**

Allowed: 0 print to syslog only
1 print to stderr and syslog

**Default:** 0

**Notes:**

Example: `-stderron 1`

**stps**

Syntax: `-stps pc1,pc2,...`

**Description:** Comma-separated list of STP point codes to which sua_if/m3ua_if should round-robin route outward messages.

**Type:** Decimal integer

**Optionality:**

Allowed: none

List of STP point codes to send messages to in round robin.

**Default:** none

**Notes:** Each PC will be substituted into the MTP destination addresses.

Example: `-stps 2,5,7`

---

### Configuring sigtran.config

**Introduction**

This topic explains the configuration options and parameters for sua_if/m3ua_if in the **sigtran.config** file.

**Note:** Unlike the parameters set in tcapif.def. These parameter cannot be set on the command line.

**sigtran.config example**

SIGTRAN TCAP Interface is installed with two **sigtran.config** files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigtran.config.example</td>
<td>A full set of parameters showing example configuration with explanations.</td>
</tr>
<tr>
<td>sigtran.config.simple</td>
<td>Two very simple configurations. This file shows minimal configuration and does not include all available parameters.</td>
</tr>
</tbody>
</table>
You can use either of these files as a source to create the sigtran.config file which will actually be used by sua_if/m3ua_if.

Note: Some specific parameters (for example host names) will need to be amended in sigtran.config to run sua_if/m3ua_if correctly.

Using eserv.config instead of sigtran.config

You can put the sigtran.config configuration into the main eserv.config file for the machine you are running sua_if/m3ua_if on. If you do this, you will need to change the ESERV_CONFIG_FILE environmental variable in the Interface's startup script. For more information about the ESERV_CONFIG_FILE environmental variable, see ESERV_CONFIG_FILE (on page 11).

Editing the file

Open the configuration file on your system using a standard text editor. Do not use text editors, such as Microsoft Word, that attach control characters. These can be, for example, Microsoft DOS or Windows line termination characters (for example: ^M), which are not visible to the user, at the end of each row. This will cause file errors when the application tries to read the configuration file.

Always keep a backup of your file before making any changes to it. This will ensure you have a working copy to which you can return.

Loading configuration changes

If you change the configuration, you must send a SIGHUP to sua_if/m3ua_if to enable the new options to take effect.

sigtran.config structure

This text shows the structure of the sigtran.config file.

```
SUA = {
  ansi = true|false
  maxSLS = int
  maxDids = int
  rejectTimeout = secs
  invokeTimers = true|false
  log = true|false
  xudtHopCount = int
  retAddrAll = true|false
  qos = int
  opc = dec int
  stpPCs = [ pc[, pc, ...] ]
  statisticsInterval = secs
  networkDebug = true|false

  connections = {
    name = {
      remote_host = [ "itp"[, "itp"] ]
      [remote_port = port]
      [local_host = [ "host"[, "host"] ]
      [local_port = port]
      [remote_role = "sg|as|as_only|*"]
      routing_context = int
      traffic_mode_type = "mode"
      [message_priority = int]
      [importance = 0|1]
      [network_indicator = 0|1]
      [network_appearance = int]
      [asp_identifier = int]
    }
  }
```

[application_server = "str"]
[transport = "sctp|tcp"]
[initiation = "str"]
[rcvbuf = bytes]
[sndbuf = bytes]
[sctp_ostreams = int]
[sctp_istreams = int]
[sctp_hbinterval = millisecs]
[sctp_init_timeout = secs]
[default_gt = "gt"]
[activate = "down|up|active"]
[segment_size = bytes]
[asp_identifier = int]
[use = "name"]
[gtt_pc = pc]
[gtt_ssn = ssn]
[gtt_remove = true|false]
[gtt_route_pc = true|false]
[gtt_np = int]

routes = [
{
[first = pc|gt last = pc|gt]
[peer = pc]
[priority = int]
[label = "str"]
[use = "str"]
[connection parameters]
[connection parameters]
}
[...]
]

classifiers = [
{
[routing_indicator = int]
[address_indicator = int]
[subsystem_number = ssn]
[point_code = pc]
[gti = int]
[trans_type = int]
[num_plan = int]
[nature_of_add = noa]
[source_routing_indicator = int]
[source_address_indicator = int]
[source_subsystem_number = ssn]
[source_point_code = pc]
[source_gti = int]
[source_trans_type = int]
[source_num_plan = int]
[source_nature_of_add = noa]

label = "str"
}
[...]
]
M3UA = {
    sigtran.config parameters
    connections = {
        other connection parameters
        mtp3_dpc = pc
    }
    [...]
    routes = [
        routes
    ]
    [classifiers = [class matches]]
}

sigtran.config parameters

The configuration variables available in sigtran.config for sua_if and m3ua_if are listed below.

ansi

Syntax: ansi = true|false
Description: Use ANSI or ITU addressing selection.
Type: Boolean
Optionality: Allowed: true ANSI addressing
false ITU addressing
Default: false
Notes: This is only used when converting a SUA address in a received packet into an SCCP address. In all other cases, the national indicator field of the address is used for that purpose.
Example: ansi = false

asn1_validate

Syntax: asn1_validate = value
Description: How to validate incoming ASN.1 data.
Type: Integer
Optionality: Optional (default used if not set).
Allowed: 0 to 3
Default: 3
Notes: Options are:
0 = no validation.
1 = validate but silently discard broken packets.
2 = validate and short syslog message for broken packets.
3 = validate and syslog broken packet contents.
Example: asn1_validate = 2

classifiers

Syntax: classifiers = [{class1}{class2}...]
Description: Defines the classification match patterns.
Type: Array
Optionality: Optional (if not set, all matching done in Route section).
Allowed:
Default: None
Notes: For more information about the configuration available for classifiers, see Classifier parameters (on page 42).
Example:

```
classifier = [
    {
        trans_type = 9
        label = "message-class-1"
    },
    {
        trans_type = 9
        num_plan = 1
        label = "message-class-2"
    }
]
```

connections

Syntax:
```
connections = {
    name = (parameters)
    [name = (parameters)]
    [...]
}
```

Description: This section defines how packets are passed from sua_if to the destination ITP/STP.

Type: Array

Optionality: Mandatory

Allowed: None

Notes: The same connection can be used for multiple routes. This removes the need to configure the same connection information each time the connection is used by a route.

Connections can inherit details from other connections using the 'use' parameter. If this is done, you can override inherited details by re-specifying a parameter within the inheriting connection.

Routes use connections by inheriting the connection details using the 'use' parameter. Like connections, you can override inherited details by specifying them within the route.

Only connections listed in the routes section are actually used.

For details of the parameters which can be used to define a connection, see Connection parameters (on page 30).

Example:
```
connections = {
    default = {
        remote_host = "supitp1"
        remote_port = 15000
        local_port = 14001
    },
    secondary = {
        use = "default"
        remote_host = "supitp2"
        asp_identifier = 1234
    }
}
```
default_retgt

Syntax: default_retgt = "str"

Description: Defines the default return Global Title which will be used when retgt_mapping is set and a match cannot be found for the return address.

Type: String
Optionality: Optional
Allowed: none

Allowed:
"1, noa, address_digits"
"2, trans_type, address_digits"
"3, trans_type, num_plan, address_digits"
"4, trans_type, num_plan, noa, address_digits"

Default: none

Notes: The variables used in the gt are:

noa Nature of Address
address_digit Destination/Called Party Number
trans_type Transmission Type
num_plan Number Plan

Example: default_retgt = "4,0,1,2,44321456"

retgt_mapping

Syntax: retgt_mapping = [
    { from="str", to = "str"},
    { from="str", to = "str"},
]

Description: Used to map return addresses. The mapping will be applied to every outgoing message. If there is no match for the return address then the value defined for default_retgt will be used.

Type: Array
Optionality: Optional
Allowed: "str" can be replaced with:
"1, noa, address_digits"
"2, trans_type, address_digits"
"3, trans_type, num_plan, address_digits"
"4, trans_type, num_plan, noa, address_digits"

Default: Defaults to mapping set in default_retgt

Notes: Variables used are:

noa Nature of Address
address_digit Destination/Called Party Number
trans_type Transmission Type
num_plan Number Plan
Example:

```
retgt_mapping = [
    { from="4,0,1,4,123", to = "4,0,1,4,441482255436"},
    { from="4,0,1,4,4414782255436", to = "4,0,1,4,441482255436"},
]
```

Note: In this example, the first entry maps the 123 address to the 441482255436 address. The second entry ensures this address is used in the future. Otherwise the setting from `default_retgt` will be used.

**invokeTimers**

Syntax: `invokeTimers = true|false`

Description: Whether to run invoke timers or not.

Type: Boolean

Optionality: 

Allowed: 

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>Run invoke timers.</td>
</tr>
<tr>
<td>false</td>
<td>Do not run invoke timers.</td>
</tr>
</tbody>
</table>

Important: Turning off invoke timers will cause extremely non-standard TCAP behavior. It should only be done in highly specialized circumstances after careful consideration of the consequences. Do not set `invokeTimers` to false without detailed consultation with Oracle about possible effects.

Default: true

Notes: For more information about invoke timers, see Q.774 3.2.1.1.3.

Example: `invokeTimers = true`

**invokeTimerOverride**

Syntax: `invokeTimerOverride = seconds`

Description: Override invoke timers with a default global value in seconds.

Type: Integer

Optionality: Optional (default used if not set).

Allowed: Valid values are 0 - 3600. See Q.774 3.2.1.1.3.

Default: 0

Notes: 

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not override invoke timers.</td>
</tr>
</tbody>
</table>

This configuration item is normally used for services that require interactions which last for >30 seconds (for example, some USSD services).

Example: `invokeTimerOverride = 0`

Warning: This configuration item is normally used for services that require interactions which last for >30 seconds (for example some USSD services).

Changing invoke timers will cause extremely non-standard TCAP behavior, and this should only be done in highly specialized circumstances after careful consideration of the consequences. Do not change `invokeTimerOverride` without detailed consultation with Oracle as to the resultant behavior.

**log**

Syntax: `log = true|false`

Description: Controls SCCP logging.

Type: Boolean
Optionality: true All SCCP level traffic is logged to /tmp.
false SCCP logging is not done.
Default: false
Notes: This may be toggled at run-time by sending a SIGUSR1 to the TCAP Interface.
SCCP logging is done at the SCCP level for both SUA and M3UA.
Example: log = false

maxDids
Syntax: maxDids = int
Description: Maximum number of in-progress transactions.
Type: Integer
Optionality: Optional
Allowed: Positive integer
Default: 100000
Notes: This parameter is not reloaded on SIGHUP.
Example: maxDids = 100000

maxSLS
Syntax: maxSLS = int
Description: Maximum Signalling Link Selection to use.
Type: Integer
Optionality: Optional
Allowed: Positive integer
Default: 255
Notes: The SLS determines the:
- Specific SS7 link that an ITP will send a packet over
- SCTP stream the TCAP Interface will use.
Example: maxSLS = 255

networkDebug
Syntax: networkDebug = true|false
Description: Whether or not to turn on the sigtran_network debug facility.
Type: Boolean
Optionality: Optional (if not specified, the sigtran_network setting from the DEBUG environment variable is used).
Allowed: true Use the sigtran_network debug.
false Use the sigtran_network setting from the DEBUG environment variable.
Default: false
Notes: The debug produces is one line per packet of debug info and some per-connection stats approximately once per second.
Example: networkDebug = true
**opc**

*Syntax:* opc = dec int

*Description:* Local point code for outgoing messages.

*Type:* Decimal integer

*Optionality:* Mandatory

*Allowed:* any valid point code

-1 Use other methods to set local point code. If you use this, you must be careful to make sure that all outgoing messages get a valid originating address.

*Note:* Only valid for SUA.

*Default:* 

*Notes:* Parameter is given as an integer.

*Example:* An ITU point code of 1-1-1 is configured as:

opc = 2057

**qos**

*Syntax:* qos = int

*Description:* SCCP/SUA protocol class.

*Type:* Decimal integer

*Optionality:* 

*Allowed:* 0 unordered delivery, no error returns

1 ordered delivery, no error returns

128 unordered delivery, return on error

129 ordered delivery, return on error

*Default:* 1

*Notes:* If you wish to have your packets randomly re-ordered, set to 0. Otherwise leave as the default 1.

*Example:* qos = 1

**rejectTimeout**

*Syntax:* rejectTimeout = secs

*Description:* Reject timeout as specified in Q.774 3.2.1.1.3.

*Type:* Integer

*Optionality:*
Allowed: 0  No timer.

**Important:** Turning off reject timers will cause extremely non-standard TCAP behavior. It should only be done in highly specialized circumstances after careful consideration of the consequences. Do not set rejectTimeout to 0 without detailed consultation with Oracle about possible effects.

Positive integer  Reject timeout.

Default: 1
Notes:
Example:  rejectTimeout = 10

retAddrAll

Syntax:  retAddrAll = true|false
Description:  Indicates how to change the return address on a received transaction.
Type:  Boolean
Optionality:  Allowed:
Default:  false  Apply strict Q.774 logic for changing the return address on a received transaction.
true  The return address may be changed at any stage of the transaction.

Default:  false
Notes:
Example:  retAddrAll = true

routes

Syntax:  routes = [  
{route1}  
{{route2}}  
[...]
  ]
Description:  Matches details in outgoing packets and sets the packets destination details.
Type:  Array
Optionality:  Mandatory
Allowed:  None
Default:  None
Notes:  Each route matches details in outgoing traffic and routes it to a specific destination (usually specified by the connection set by the use parameter). Matching details include one of the following:
- A range of point-codes (numeric)
- A range of GT prefixes (strings)
If two (or more) routes have identical ranges, they are merged and all the endpoints used. If one range is strictly contained within another, the narrower range overrides the larger.
Outgoing traffic may also match details in the classifiers section, which will then use the route with the same label to define the destination.
For more information about the configuration available for routes, see Route
parameters (on page 40).

Example:

```cpp
routes = [
    {
        first = 0
        last = 16777215
        use = "default"
    },
    {
        first = 0
        last = 16777215
        use = "secondary"
    }
]
```

**scmg_ssn**

**Syntax:** `scmg_ssn = int`

**Description:** The sub-system number for SCMG processing by the M3UA interface.

**Type:** Integer

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

**Allowed:** Integer value between 0 and 255

**Default:** No SCMG processing will take place.

**Notes:**

**Example:** `scmg_ssn = 1`

**statisticsInterval**

**Syntax:** `statisticsInterval = secs`

**Description:** How often in seconds to notify statistics to syslog.

**Type:** Decimal integer

**Optionality:**

**Allowed:**
- 0 turn off notification
- positive integer notification period

**Default:** 0

**Notes:**

**Example:** `statisticsInterval = 0`

**stpPCs**

**Syntax:** `stpPCs = [ stp1, stp2, ... ]`

**Description:** Failover routing to the routes configured in the routes = [] section.

**Type:** Decimal integer

**Optionality:** Optional

**Allowed:**

**Default:**

**Notes:**

The routing logic follows these steps:

1. Attempt to route a packet directly to its real destination address.
2. If that fails, choose an STP point code from the stpPCs list, and route using that point code.

This gives two separate mechanisms for spreading traffic over multiple destinations. The stpPCs list is more limited (because it applies to all
destinations) but it is integrated with TCAP dialogs (it ensures multiple packets on 
the same TCAP dialog use the same STP PC).
The STP list can also be specified on the command line. The command line 
configuration will override any set here.
The whitespace used within the square brackets does not affect the parsing of the 
STP references.
For more information about TCAP dialogs, see SLEE Technical Guide.

Example:
```
stpPCs = [ 1234, 1235 ]
```

xudtHopCount

Syntax: `xudtHopCount = int`

Description: The hop count to send on protocols that support it.

Type: Decimal integer

Optionality: Allowed: -1 Do not set a hop count
1-255 Hop count

Default: -1

Notes: For SCCP over M3UA, we send UDTs if the hop count is -1, and XUDTs 
otherwise.

Example: `xudtHopCount = -1`

Connection key configuration

These parameter values identify which connection a route uses.

- local_host
- local_port
- remote_host
- remote_port
- transport

Connection parameters

The connections subsection of the `sigtran.config` configuration supports these parameters.

```python
connections = {
  name = {
    remote_host = [ "itp", "itp"]
    [remote_port = port]
    [local_host = [ "host", "host"]]
    [local_port = port]
    [remote_role = "sg|as|as_only|*"]
    routing_context = int
    traffic_mode_type = "mode"
    [message_priority = int]
    [importance = 0|1]
    [network_indicator = 0|1]
    [network_appearance = int]
    [asp_identifier = int]
    [application_server = "str"]
    [transport = "sctp|tcp"]
    [initiation = "str"]
    [rcvbuf = bytes]
    [sndbuf = bytes]
  }
}
```
The parameters in this subsection are described in detail below.

activate
Syntax: `activate = "down|up|active"`
Description: Set the target state for a connection.
Type: String
Optionality: Optional
Allowed: `down` Connection is disabled.
`up` Connection will change to UP state (but not ACTIVE). Remote attempts to activate will get a management-blocking error.
`active` Connection will be allowed to change to ACTIVE state following the normal process.
Default: active
Notes: Override default by specifying a name for the AS to which some ASPs belong.
Example: `activate = "active"

application_server
Syntax: `application_server = "str"`
Description: The remote peers may be grouped into application servers for override handling. If the peers are override, then we try and maintain only one ASP active in each AS at any one time.
Type: String
Optionality: Optional
Allowed: Any name for the AS to which the ASPs belong.
Default: By default, the interface attempts to guess the ASP to AS groupings from the routing information.
Notes: Override default by specifying a name for the AS to which some ASPs belong.
Example: `application_server = "appserver1"

asp_identifier
Syntax: `asp_identifier = int`
Description: The ASP Identifier sent to the SGP.
Type: Decimal integer
<table>
<thead>
<tr>
<th>Optionality:</th>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allowed:</td>
<td></td>
</tr>
<tr>
<td>Default:</td>
<td></td>
</tr>
<tr>
<td>Notes:</td>
<td>This is an arbitrary number expected by the remote entity.</td>
</tr>
<tr>
<td>Example:</td>
<td>asp_identifier = 5</td>
</tr>
</tbody>
</table>

**default_gt**

**Syntax:**
```
default_gt = "gt"
```

**Description:** Override outgoing calling party address with route on PC/SSN to one of the following:
- Route on GT
- Using the specified GT

**Type:** Numeric string
**Optionality:** Optional
**Allowed:**
**Default:** The default is to not set this parameter (leave PC/SSN routing alone).
**Notes:** For most purposes, the retgt (on page 16) command line parameter will provide more predictable behaviour.
**Example:**
```
default_gt = "12345678"
```

**gtt_pc**

**Syntax:**
```
gtt_pc = pc
```

**Description:** Sets point code for Global Title Translation parameters used to set a PC on the destination address.

**Type:** Decimal integer
**Optionality:** Optional
**Allowed:**
**Default:**
**Notes:** Used for routing an outgoing packet on GT.
The PC & SSN supplied by GT only effect the packet content - it does not change where we send it. The GTT will not override a PC and SSN already on the address.
The GTT parameters are ignored when routing on PC/SSN.
**Example:**
```
gtt_pc = 1234
```

**gtt_np**

**Syntax:**
```
gtt_np = int
```

**Description:** Sets NP for Global Title Translation parameters used to set a PC and SSN on the destination address.

**Type:** Integer
**Optionality:** Optional
**Allowed:**
**Default:**
**Notes:**
gtt_np has no effect if gtt_remove is set, or if the GT type does not have an NP.
ITU GTI 3 and 4 have NP.
ITU GTI 1 and 2 do not have NP.

**Example:**
gtt_remove
Syntax:  
gtt_remove = true|false
Description: Defines GT details in outgoing message using GT routing.
Type: Boolean
Optionality: Optional
Allowed: true Remove GT and replace with PC.
false Leave existing GT.
Default: false
Example: gtt_remove = true


gtt_route_pc
Syntax: 
gtt_route_pc = true|false
Description: Whether or not to override routing indicator.
Type: Boolean
Optionality: Optional
Allowed: true Route on PC.
false Do not change routing indicator.
Default: false
Example: gtt_route_pc = true


gtt_ssn
Syntax:  
gtt_ssn = ssn
Description: Sets Subsystem Number for Global Title Translation parameters used to set an SSN on the destination address.
Type: Hex integer
Optionality: Optional
Allowed: 
Default: 
Notes: Used for routing an outgoing packet on GT.
The PC & SSN supplied by GTT only effect the packet content - it does not change where we send it. The GTT will not override a PC and SSN already on the address.
The GTT parameters are ignored when routing on PC/SSN.
Example: gtt_ssn = 11

importance
Syntax:  
importance = 0|1
Description: Sets the SCCP XUDT/LUDT and SUA importance field.
Type: Boolean
Optionality: Optional
Allowed: 0 Sets field to 0 (normal)
1 Sets field to 1 (important)
Default: 0
Notes:
Example: importance = 0

**initiation**

**Syntax:** initiation = "str"
**Description:** Overrides the default action for local and remote ports.
**Type:** String
**Optionality:** Optional
**Allowed:**
- listen Listen for incoming connections.
- connect Attempt to connect to remote peer.
- none Do not listen or attempt to connect.
- both Listen for incoming connections and attempt to connect to remote peer.

**Default:** By default, if the local_port is specified, TCAP Interface listens for incoming connections, and if the remote_port is given, TCAP Interface attempts to connect.

**Notes:** In some circumstances the default behavior may be undesirable.

**Examples:**
- Where the remote end must use a fixed port number but we want the remote end to initiate the connection).
- When using SIGTRAN over TCP, listening on a port means that you can't later connect using that local port.

**Example:** initiation = "connect"

**local_host**

**Syntax:** Either:
- local_host = "localAddr"
- local_host = [ "localAddr1", "localAddr2", ... ]

**Description:** IP address or hostname of local host.
**Type:** String
**Optionality:** Optional
**Allowed:** Any valid:
- hostname
- hostname and domain
- ip address

**Default:** Using an array of remote hosts provides control over the SCTP multi-home functionality. The local_host list is used to bind the local socket.

If you leave the local_host unset, the SCTP will enable multi-homing over all addresses.

The connections are identified internally by only the first remote_host.

**Examples:**
- local_host = "produas01.telcoexample.com"
- local_host = [ "localAddr1", "localAddr2" ]

**local_port**

**Syntax:** local_port = port
**Description:** Local host port number.
Chapter 2

Type: Decimal integer
Optionality: At least one of remote_port and local_port must be specified. If using SCTP, both may be specified.
Allowed:
Default: If not set, the Operating System will set a port for outgoing messages.
Notes: Example: local_port = 14001

message_priority
Syntax: message_priority = int
Description: Sets the MTP3 message_priority field in outgoing M3UA packets.
Type: Decimal integer
Optionality: Optional
Allowed:
Default: 0
Notes: If not set, is set to zero in M3UA packets and omitted from SUA packets.
Example: message_priority = 0

network_appearance
Syntax: network_appearance = int
Description: M3UA network appearance value sent in M3UA packets sent to the network.
Type: Integer
Optionality: Optional
Allowed:
Default: 0
Notes: If not set, this field is omitted from M3UA packets sent to the network.
Example: network_appearance = 10

routing_context
Syntax: routing_context = int
Description: The SIGTRAN routing context.
Type: Decimal integer
Optionality: Generally mandatory for SUA.
Generally optional for M3UA.
Allowed: 32-bit integers
Default:
Notes: The context is derived from the ACTIVE handshake if possible.
Example: routing_context = 666

network_indicator
Syntax: network_indicator = 0|2
Description: Sets the value of the MTP3 network indicator in outgoing messages.
0000 International
0010 National
Type: A 4-bit binary field
Optionality:   Mandatory
Allowed:   0, 2
Default:  0
Notes:  As defined by ANSI / ITU-T
Example: network_indicator = 0

rcvbuf

Syntax:  \texttt{rcvbuf} = \texttt{bytes}
Description:  Receive buffer size.
Type:  Decimal integer
Optionality:  Optional
Allowed:  
  \begin{itemize}
    \item positive integer  Set the rcvbuf value in outgoing
    \item Use the OS default.
  \end{itemize}
Default:  0
Notes:  \begin{itemize}
    \item \textbf{Important:} This value needs to cover all outstanding unacknowledged SCTP data
    \item on the network. If you set these to a non-zero value, it is unlikely to work properly
    \item if set to less than 64000.
    \item Do not change this parameter from the default value without calculating the new
    \item value based on network conditions. The defaults should work fine in all but the
    \item highest traffic conditions.
  \end{itemize}
Example:  rcvbuf = 131072

remote_host

Syntax:  Either:
  \begin{itemize}
    \item remote_host = \texttt{"itpAddr"}
    \item remote_host = [ \texttt{"itpAddr1"}, \texttt{"itpAddr2"}, ... ]
  \end{itemize}
Description:  Location of the remote host(s).
Type:  String or array of comma-separated strings.
Optionality:  Mandatory
Allowed:  Any valid:
  \begin{itemize}
    \item hostname
    \item hostname and domain
    \item ip address
  \end{itemize}
Default:  
Notes:  Using an array of remote hosts provides control over the SCTP multi-home functionality. When
  connecting, remote_host gives the candidate peer addresses. When listening, and incoming connection
  matches if it's primary IP address matches any of the remote_hosts.
  The connections are identified internally by only the first remote_host.
Examples:  
  remote_host = "itp"
  remote_host = [ "itp1", "itp2" ]

remote_port

Syntax:  \texttt{remote_port} = \texttt{port}
Description:  Port to connect to on remote_host.
Chapter 2

remote_port
Type: Decimal integer
Optionality: At least one of remote_port and local_port must be specified. If using SCTP, both may be specified.
Allowed: If not set, the operating system will accept connections from any remote port.
Default:
Notes: Example: remote_port = 26600

remote_role
Syntax: remote_role = "sg|as|as_only|"
Description: How UPs and ACTIVES are expected to flow between TCAP Interface and peer.
Type: String
Optionality: Optional, default is used if not set.
Allowed: sg Peer is expected to receive UPs and ACTIVES but not send them to us.
       as UPs and ACTIVES should go both ways.
       as_only Peer is expected to send UPs and ACTIVES but we will not send them to the peer.
       * Both ways are attempted, but only one direction needs to be active for data to flow.
Default: ***
Notes: *** should work in most circumstances.
Example: remote_role = "sg"

sctp_hbinterval
Syntax: sctp_hbinterval = millisecs
Description: SCTP heartbeat interval in milliseconds.
Type: Decimal integer
Optionality: Optional (default used if not set).
Allowed: 0 Turn off heartbeat messages.
         positive integer Heartbeat interval.
Default: 1000
Notes: To speed up failover behavior, it may help to reduce some kernel-level parameters using the ndd command, including:
       • sctp_pp_max_retr
       • sctp_rto_min
       • sctp_rto_max
       • sctp_rto_initial.
Example: sctp_hbinterval = 2500

sctp_init_timeout
Syntax: sctp_init_timeout = secs
Description: Gives the maximum time the SIGTRAN stack will allow for an SCTP association to form. If the association fails to form within this time the attempt is abandoned and a new attempt is begun.
Type: Decimal integer
**Optionality:** Optional (default used if not set).

**Allowed:**
- **0**
  - No timeout will be applied and the operating system's values are all that will be used.
- **positive integer**
  - Length of timeout, in seconds

**Default:** 5

**Notes:**
Note that this interacts very heavily with the operating system's SCTP timeout values. Its purpose is to ensure that the SIGTRAN stack will attempt to form a connection at least every time the timeout fires. The default of five seconds means that no matter how the operating system is configured the SIGTRAN stack will attempt to connect every five seconds. This is very useful in case the operating system has not been configured or has been incorrectly configured.

In Solaris there are several important configuration items related to this. The key ones are `sctp_max_init_retr`, which determines how many times the kernel will send the SCTP INIT message (the initial message plus this number of retries) and `sctp_rto_max` which determines the maximum retry time. Solaris uses an exponential backoff system as it retries.

Setting `sctp_rto_max` to 1000 (the units are milliseconds) will ensure that the connection is retried every second thus giving us the fastest restart time.

**Example:**

```plaintext
sctp_init_timeout = 5
```

**sctp_istreams**

**Syntax:**

```
sctp_istreams = int
```

**Description:** Number of SCTP input streams per connection.

**Type:** Decimal integer

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

**Allowed:**
- **0**
  - Use the OS default.
- **positive integer**
  - Maximum number of SCTP input streams per connection.

**Default:** 0

**Notes:**

**Example:**

```plaintext
sctp_istreams = 10
```

**sctp_ostreams**

**Syntax:**

```
sctp_ostreams = int
```

**Description:** Number of SCTP output streams per connection.

**Type:** Decimal integer

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

**Allowed:**
- **0**
  - Use the OS default.
- **positive integer**
  - Maximum number of SCTP output streams per connection.

**Default:** 0

**Notes:**

**Example:**

```plaintext
sctp_ostreams = 10
```

**segment_size**

**Syntax:**

```
segment_size = bytes
```

**Description:** The maximum number of bytes allowed in a segment in an SCCP payload.

**Type:** Decimal integer
Optionality: Optional

Allowed: If not set, segmentation will not be performed.

Default: Over SCCP/M3UA, only used if xudtHopCount is set (so that XUDT packets are sent). If sudtHopCount is not set, then UDT packets are sent, and segment_size is ignored.

Notes: If set, and network is SUA, segment_size is always applied.

Example: segment_size = 255

sndbuf

Syntax: \[ \text{Description: Send buffer size, in bytes.} \]
Type: Decimal integer
Optionality: Optional

Allowed: 0 Use the OS default.

Notes: Important: This value needs to cover all outstanding unacknowledged SCTP data on the network. If you set these to a non-zero value, it is unlikely to work properly if set to less than 64000.

Do not change this parameter from the default value without calculating the new value based on network conditions. The defaults should work fine in all but the highest traffic conditions.

Example: sndbuf = 131072

traffic_mode_type

Syntax: \[ \text{traffic_mode_type = "mode"} \]
Description: Traffic mode type.
Type: String
Optionality: Optional. However, if not set, sua_if/m3ua_if will be using loadshare and peers may not be, which will result in ITPs failing.

Allowed: override SIGTRAN override, sua_if/m3ua_if will wait for a request before setting a connection to active.

override_primary SIGTRAN override, sua_if/m3ua_if will set connections to active as soon as possible.

loadshare sua_if/m3ua_if will send messages on sequential connections in a round robin (one after the other) pattern.

broadcast sua_if/m3ua_if will send messages on all applicable connections.

Default: If not specified or negotiated, loadshare is used.

Notes: Example: traffic_mode_type = "loadshare"
transport
Syntax: transport = "sctp|tcp"
Description: The transport type for this connection.
Type: String
Optionality: Optional
Allowed: sctp SIGTRAN over SCTP
tcp SIGTRAN over TCP
Default: sctp
Notes: In normal networks, SCTP is always used.
Example: transport = "stcp"

use
Syntax: use = "name"
Description: Base this connection configuration on the named connection.
Type: String
Optionality: Optional
Allowed: Any already defined connection name.
Default: 
Notes: This enables parameters to be redefined specifically for this connection without the need to repeat the unchanged ones.
Example: use = "PrimaryConnection"

Route parameters
The routes subsection of the sigtran.config configuration supports these parameters.

Notes:
- All routes have the same parameters available as listed for Connection parameters (on page 30), plus those listed below for Route parameters.
- Routes can also use parameters from the Classifier parameters (on page 42) section, though this is only recommended for simple routing situations.

routes = [
  {
    [first = pc|gt last = pc|gt]
    [peer = pc]
    [priority = int]
    [label = "str"]
    [use = "str"]
    [connection parameters]
    [connection parameters]
  }
  [...]
]
The parameters in this subsection are described in detail below.

first
Syntax: Either:
  - first = startPC
  - first = "startGT"
Description: The first number (Point Codes) or string (Global Title) in the range of destination
addresses.

Type: Decimal integer or Numeric string
Optionality: Optional. Must be set if last is set. Must be set if peer is not set. Must not be set if peer is set.
Allowed: 
Default: 
Notes: This number is inclusive.
Examples: 
first = 0
first = "6441"

label

Syntax: label = "str"
Description: If a message has matched a classifier with the same label value as this parameter, the message can use this route.
Type: String
Optionality: Optional (not used if not set).
Allowed: None (no label).
Default: 
Notes: If this route will be used by messages which have been matched in the Classifier parameters (on page 42) section, this string must match a label value exactly.
If more than one route has the same label, qualifying messages will follow the usual routing match rules to determine which route to use.
Example: label = "itp-special"

last

Syntax: Either:

- last = endPC
- last = "endGT"
Description: The last Point Code or Global Title prefix in the range of destination addresses.
Type: Decimal integer or Numeric string
Optionality: Optional. Must be set if first is set. Must be set if peer is not set. Must not be set if peer is set.
Allowed: 
Default: 
Notes: This number is inclusive.
Examples: 
last = 6441
last = "6441"

peer

Syntax: peer = pc
peer = "gt"
Description: Point Code or Global Title prefix of the application server to connect to. Routes to a single point code instead of using the first and last parameters.
Type: Point codes are decimal integer.
Global Title prefixes are string.
Optionality: Optional. Must be set if first or last are not set. Must not be set if first or last are
set.

Allowed: None.
Default: None.
Notes: peer = "all_pc" supplies a route covering all point codes.
        peer = "***" supplies a route covering all point codes and all global titles.
Examples: peer = 131330
          peer = "64321"

priority
Syntax: priority = priority
Description: The route's priority.
Type: Decimal integer
Optionality: Required
Allowed: 0
Default: None.
Notes: The usable routes with the highest priority will be used for a message.
       If multiple routes with the same priority are given, they are round-robined (unless a remote ASP specifically requests override mode).
       The priority is per route, not per-connection, so a priority 1 for a route is not shared with the other routes using the same connection (although an item in the ‘connections’ section above may be given a priority to be used unless overridden).
Example: priority = 1

use
Syntax: use = "route_name"
Description: Base this route configuration on the named route.
Type: String
Optionality: Optional
Allowed: Any already defined route name.
Default:
Notes: This enables parameters to be redefined specifically for this route without the need to repeat the unchanged ones.
Example: use = "PrimaryRoute"

Classifier parameters
The classifiers section is used to set up routing based on things other than the destination PC or global-tile digits. If one of the classifiers matches a message, then routes with the same 'label' as the classifier are used.

The classifiers are tested in order, and the first matching classifier is used. To match, a message must match all the conditions. If no classifiers match, then the routes with no label parameter are used.

This section is optional. The classifier-matches may be inserted directly into the 'routes' section. For more information about this, see Route parameters (on page 40).

```plaintext
classifiers = [
    { [routing_indicator = int] [address_indicator = int] [subsystem_number = ssn] }
```
[point_code = pc]
[gti = int]
[trans_type = int]
[num_plan = int]
[nature_of_add = noa]

[source_routing_indicator = int]
[source_address_indicator = int]
[source_subsystem_number = ssn]
[source_point_code = pc]
[source_gti = int]
[source_trans_type = int]
[source_num_plan = int]
[source_nature_of_add = noa]

label = "str"
}
[...]
]
The parameters in this subsection are described in detail below.

address_indicator

Syntax: address_indicator = int
Description: For SUA, match against the address indicator field from the destination SUA address.
For SCCP, match against the address indicator field from the destination SCCP address (the first byte of the SCCP address).
Type: Integer
Optionality: Optional, (if not set, no matching is done against address indicator).
Allowed: None.
Default: None.
Notes: Example: address_indicator = 2

gti

Syntax: gti = int
Description: For SUA, match against the GTI field from the destination SUA address.
For SCCP, match against the GTI field from the destination SCCP address (the middle 4 bits of the address indicator byte).
Type: Integer
Optionality: Optional, (if not set, no matching is done against GTI).
Allowed: For SCCP, only 0 through 4 are used.
Default: None.
Notes: Example: gti = 2

label

Syntax: label = "str"
Description: Traffic which matches a classifier with this label, will be routed to the route which also specified this label.
Type: String
Chapter 2

Optionality: Mandatory

Allowed:

Default: None.

Notes: Must be set for each classifier entry. If this value is not set, the traffic will route down the default route.

Example: label = "itp-special"

**nature_of_add**

Syntax: nature_of_add = noa

Description: Match against the NOA field from the destination SUA or SCCP address.

Type: Integer

Optionality: Optional, (if not set, no matching is done against NOA).

Allowed:

Default: None.

Notes:

Example: nature_of_add = 1

**num_plan**

Syntax: num_plan = int

Description: For SUA, match against the Number Plan field from the destination SUA address. For SCCP, match against the Numbering Plan in the destination SCCP address (the top four bits of the GT numbering plan / encoding scheme byte).

Type: Integer

Optionality: Optional, (if not set, no matching is done against number plan).

Allowed: For SCCP, 0 - 15

Default: None.

Notes:

Example: num_plan = 2

**point_code**

Syntax: point_code = pc

Description: Match against the PC field from the destination SUA/SCCP address.

Type: Integer

Optionality: Optional, (if not set, no matching is done against point code).

Allowed:

Default: None.

Notes:

Example: point_code = 1234

**routing_indicator**

Syntax: routing_indicator = int

Description: What details in the destination SUA/SCCP address to match against.

Type: Integer

Optionality: Optional, (if not set, no matching is done against routing indicator).

Allowed: 0 Route on GT

1 Route on PC and SSN.
Default: None.
Notes: 
Example: routing_indicator = 1

**subsystem_number**

Syntax: subsystem_number = ssn
Description: Match against the SSN field from the destination SUA/SCCP address.
Type: Integer
Optionality: Optional, (if not set, no matching is done against SSN).
Allowed: 
Default: None.
Notes: 
Example: subsystem_number = 8

**trans_type**

Syntax: trans_type = int
Description: Match against the translation type field from the destination SUA/SCCP address.
Type: Integer
Optionality: Optional (if not set, no matching is done against translation type).
Allowed: Any valid translation type.
Default: None.
Notes: 
Example: trans_type = 9

**M3UA route parameters**

The M3UA configuration is almost identical to the SUA config. The only difference is the additional parameters listed here.

The connections subsection of the sigtran.config configuration supports these parameters.

```plaintext
M3UA = {
  sigtran.config parameters
  connections = {
    [other connection parameters
      mtp3_dpc = pc
    ]
    [...]
  }
  routes = {
    routes
    [classifiers = [class matches]]
  }
}
```

The parameters in this subsection are described in detail below.

**mtp3_dpc**

Syntax: mtp3_dpc = pc
Description: The destination point code required in MTP3 header.
Type: Decimal integer
Optionality:
Allowed:
Default:
Notes: There are three possible sources:
- The SCCP address
- The gtt_pc (when routing on GT)
- Explicitly configured

Example:
\[ \text{mtp3_dpc} = 3245 \]

Example Configuration Scenarios

Example - route 2 PCs

This is a small and simple \texttt{eserv.config} for the SUA stack. This example routes to 2 PCs. However, because they are configured as STP PCs, traffic is re-directed to STPs. All end-points have two IP addresses, so SCTP multi-homing is used.

\[
\text{SUA} = \{
    \# Local point code.
    \text{opc} = 2057
    \# Failover list for failed traffic.
    \text{stpPCs} = [ 4101, 4102 ]
    \text{connections} = \{
        \text{itp1} = \{
            \# multi-home to both nics.
            \text{remote_host} = [ "itp1-nicA", "itp1-nicB" ]
            \text{remote_port} = 14001
            \text{local_host} = [ "local-nicA", "local-nicB" ]
            \# specify routing context and loadshare for ITPs.
            \text{routing_context} = 666
            \text{traffic_mode_type} = "loadshare"
        };
        \text{itp2} = \{
            \# multi-home to both nics.
            \text{remote_host} = [ "itp2-nicA", "itp2-nicB" ]
            \text{remote_port} = 14001
            \text{local_host} = [ "local-nicA", "local-nicB" ]
            \# specify routing context and loadshare for ITPs.
            \text{routing_context} = 666
            \text{traffic_mode_type} = "loadshare"
        };
    \}
    \text{routes} = [\{
        \text{peer} = 4101 \# Route all traffic to peer 4101 to
        \text{use} = "itp1" \# dest defined in connection 'itp1'.
    \},\{
        \text{peer} = 4102 \# Route all traffic to peer 4102 to
        \text{use} = "itp2" \# dest defined in connection 'itp2'.
    \}];
\}
\]
Diagram - route 2 PCs

Here is a diagram which shows some of the features of the example.

Example - route all to balanced ITPs

This example directly routes all PCs and all GTs to a pair of load-balanced ITPs (itp1 and itp2).

SUA = {
    opc = 2057
    connections = {
        multi-home = {
            # multi-home to both itps.
            remote_host = [ "itp1", "itp2" ]
            remote_port = 14001
            local_host = [ "localhost" ]
            routing_context = 666
            traffic_mode_type = "loadshare"
        }
    }
    routes = [
        {
            peer = "**"           # Route all traffic to
            use = "multi-home"   # destns in connection 'multi-home'.
        }
    ]
}

Note: The STP mechanism is not used, so the TCAP transactions are not tracked (as there is only one destination, that doesn't matter).
Diagram - route all to balanced ITPs

Here is a diagram which shows some of the features of the example.

Example - route on transmission type

This example directly routes all PCs and all GTs to an ITP (itp1), except for traffic with transmission type 9 which goes to itp3.

```
SUA = {
    opc = 2057

    connections = {
        multi-home = {
            # multi-home to both itps.
            remote_host = ["itp1", "itp2"]
            remote_port = 14001
            local_host = ["localhost"]
            routing_context = 666
            traffic_mode_type = "loadshare"
        }
        itp3 = {
            remote_host = "itp3" # Send to itp3.
            use = "multi-home"   # Use main config to set defaults.
        }
    }

    routes = [
        {
            peer = "*"          # Route all traffic to
            use = "multi-home"  # destns in connection 'multi-home'.
        }
        {
            label = "tt9"       # Class 'tt9' will use
            use = "itp3"        # dest defined in connection 'itp3'.
        }
    ]
}
```

```
classifiers = [
    {
        trans_type = 9     # Route all traffic with transmission type 9
        label = "tt9"      # to dest in route labelled 'tt9'
    }
]```
Diagram - route on transmission type

Here is a diagram which shows some of the features of the example.

Routes:
All traffic, use connection 'multi-home'. Except transmission type 9, use connection 'itp3'.

pc=2057
localhost

collection
multi-home

pc=4101
itp1, itp2
port=14001

connection

itp1

pc=4101
itp3
port=14001

itp3
Overview

Introduction

This chapter explains the processes which run automatically as part of the application. These processes are started automatically by one of the following:

- inittab
- crontab
- Service Logic Execution Environment SLEE

Note: This chapter also includes some plug-ins to background processes which do not run independently.

In this chapter

This chapter contains the following topics.

sua_if 51
m3ua_if 52

sua_if

Purpose

sua_if is the main binary in the SIGTRAN TCAP Interface, for a SUA installation.

Startup

sua_if is an SLEE Interface and is started during SLEE initialization. The line in the SLEE.cfg which starts the sua_if is:

INTERFACE=sua_if sua_if.sh /IN/service_packages/SLEE/bin EVENT

For instructions about starting and stopping sua_if, see SLEE Technical Guide.

Configuration

sua_if is configured by the:

- File set by the ESERV_CONFIG_FILE environmental variable (usually this will be sigtran.config)
- Command line parameters that can also be set in tcapif.def

For more information about configuring this binary, see Configuration Overview (on page 9).

Output

The sua_if process writes error messages to the system messages file. It also writes additional output to the /IN/service_packages/SLEE/tmp/sigtran.log file.
m3ua_if

Purpose

m3ua_if is the main binary in the SIGTRAN TCAP Interface, for a M3UA installation.

Startup

m3ua_if is an SLEE Interface and is started during SLEE initialization. The line in the SLEE.cfg which starts the m3ua_if is:

```
INTERFACE=m3ua_if m3ua_if.sh /IN/service_packages/SLEE/bin EVENT
```

For instructions about starting and stopping m3ua_if, see SLEE Technical Guide.

Configuration

m3ua_if is configured by the:

- File set by the ESERV_CONFIG_FILE environmental variable (usually this will be sigtran.config)
- Command line parameters that can also be set in tcapif.def.

For more information about configuring this binary, see Configuration Overview (on page 9).

Output

The sua_if process writes error messages to the system messages file. It also writes additional output to the /IN/service_packages/SLEE/tmp/sigtran.log file.
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.

Common Troubleshooting Procedures

Debug

53

Common Troubleshooting Procedures

Introduction

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

Debug

Introduction

sua_if/m3ua_if contains a logging capability that can be used to store messages received and sent by the stack.

Messages are recorded at SCCP-level for both SUA and M3UA.

Setup

To turn on the SCCP-level message logging, set the log parameter to true, and reload the configuration.

Note: You can reload the configuration by sending the interface a SIGHUP.

For more information about the log parameter, see log (on page 25).

Output

Once logging is switched on the interface will append to a binary log file in /IN/service_packages/SLEE/tmp.

Output is in the form of raw data which can be decoded with the tcread program.
Overview

Introduction

This chapter explains the statistics produced by the application, and the reports you can run on the statistics.

In this chapter

This chapter contains the following topics.

Statistics

Introduction

The SIGTRAN TCAP Interface logs an array of statistics counters for each socket. Statistics are kept for:

- Errors
- Packets sent and received (per packet type)

Statistics for all connections

This table describes the statistics which are kept for all connections:

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RX_total</td>
<td>Total packets received.</td>
</tr>
<tr>
<td>TX_total</td>
<td>Total packets sent.</td>
</tr>
<tr>
<td>RX_error</td>
<td>Number of errors when attempting to read a packet from a connection.</td>
</tr>
<tr>
<td>TX_error</td>
<td>Number of errors when attempting to write a packet to a connection.</td>
</tr>
<tr>
<td>RX_corrupt</td>
<td>Number of packets received that contain incorrect data. This may be:</td>
</tr>
<tr>
<td></td>
<td>• Incorrectly formatted packets</td>
</tr>
<tr>
<td></td>
<td>• Correctly formatted packet that is not valid for the current state of the connection</td>
</tr>
<tr>
<td>TX_congest</td>
<td>Number of outbound packets dropped due to transmit buffers being full.</td>
</tr>
<tr>
<td>connection_failures</td>
<td>Number of times an error has resulted in the connection being disconnected.</td>
</tr>
</tbody>
</table>

SUA connections

For each packet type (for example, CLDT) two statistics are kept with the:
• RX_ prefix, for packets received (for example, RX_CLDT).
• TX_ prefix, for packets sent (for example, TX_CLDT).

**M3UA connections**

For each packet type (for example: DATA) two statistics are kept with the:

- RX_ prefix, for packets received (for example, RX_DATA).
- TX_ prefix, for packets sent (for example, TX_DATA).

**SCCP over M3UA connections**

In addition to the M3UA statistics, for each SCCP packet type (for example: UDT) two statistics are kept with the:

- RX_SCCP_ prefix, for packets received (for example, RX_SCCP_UDT).
- TX_SCCP_ prefix, for packets sent (for example, TX_SCCP_UDT).

**Note:** A SCCP UDT packet contained in a M3UA DATA packet will be counted in both the relevant SCCP and M3UA counters.
Chapter 6

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.

Installation and Removal Overview
Checking the Installation

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

SIGTRAN packages

An installation of SIGTRAN includes the following packages, on the:

- SMS:
  - sigtranSms
- SLC:
  - sigtranScp

Checking the Installation

Introduction

Refer to these checklists to ensure that SIGTRAN TCAP Interface has installed correctly.

Checklist - SMS

Follow the steps in this checklist to ensure SIGTRAN TCAP Interface has been installed on a SMS machine correctly.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log into the system as acs_op.</td>
</tr>
</tbody>
</table>
Step | Action
--- | ---
2 | Enter `sqlplus /`
   
   **Note:** No password is required.
3 | Ensure that the SMF_STATISTICS_DEFN table in the SMF database has been updated to include SIGTRAN TCAP Interface entries.

**Checklist - non-SMS**

Follow the steps in this checklist to ensure SIGTRAN TCAP Interface has been installed on a non-SMS machine correctly.

<table>
<thead>
<tr>
<th>Step</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Log into the machine as the user which operates the SLEE running on this machine.</td>
</tr>
<tr>
<td>2</td>
<td>Check the <code>/IN/service_packages/SLEE</code> directory structure exists with subdirectories.</td>
</tr>
</tbody>
</table>
| 3 | Check the `/IN/service_packages/SLEE/bin` directory contains the following files:
   - `sua_if`
   - `sua_if.sh`
   - `m3ua_if`
   - `m3ua_if.sh` |
| 4 | Check the `/IN/service_packages/SLEE/etc` directory contains the following files:
   - `sigtran.conf.example`
   - `sigtran.conf.sample` |

**Process list**

If the application is running correctly, the following processes should be running on each non-SMS machine:

- Started during SLEE startup, one of the following:
  - `sua_if`
  - `m3ua_if`
NCC Glossary of Terms

AAA

ACS
Advanced Control Services configuration platform.

AS
Application Server. The logical entity serving a SUA routing key. An AS is equivalent to an SS7 end point (for example, HLR, MSC, …). An AS contains, at least, one ASP.

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.

ASP
- Application Service Provider, or

CAMEL
Customized Applications for Mobile network Enhanced Logic
This is a 3GPP (Third Generation Partnership Project) initiative to extend traditional IN services found in fixed networks into mobile networks. The architecture is similar to that of traditional IN, in that the control functions and switching functions are remote. Unlike the fixed IN environment, in mobile networks the subscriber may roam into another PLMN (Public Land Mobile Network), consequently the controlling function must interact with a switching function in a foreign network. CAMEL specifies the agreed information flows that may be passed between these networks.

CAP
CAMEL Application Part

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

cron
Unix utility for scheduling tasks.

crontab
File used by cron.

Diameter
A feature rich AAA protocol. Utilises SCTP and TCP transports.
DP
Detection Point

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.

GPRS
General Packet Radio Service - employed to connect mobile cellular users to PDN (Public Data Network- for example the Internet).

GSM
Global System for Mobile communication.
It is a second generation cellular telecommunication system. Unlike first generation systems, GSM is digital and thus introduced greater enhancements such as security, capacity, quality and the ability to support integrated services.

GT
Global Title.
The GT may be defined in any of the following formats:
- Type 1: String in the form "1,<noa>,<BCD address digits>"
- Type 2: String in the form "2,<trans type><BCD address digits>"
- Type 3: String in the form "3,<trans type>,<num plan>,<BCD address digits>"
- Type 4: String in the form "4,<trans type>,<num plan>,<noa>,<BCD address digits>"
The contents of the Global Title are defined in the Q713 specification, please refer to section 3.4.2.3 for further details on defining Global Title.

HLR
The Home Location Register is a database within the HPLMN (Home Public Land Mobile Network). It provides routing information for MT calls and SMS. It is also responsible for the maintenance of user subscription information. This is distributed to the relevant VLR, or SGSN (Serving GPRS Support Node) through the attach process and mobility management procedures such as Location Area and Routing Area updates.

HPLMN
Home PLMN

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.

IDP
INAP message: Initial DP (Initial Detection Point)
IN
Intelligent Network

INAP
Intelligent Network Application Part - a protocol offering real time communication between IN elements.

Initial DP
Initial Detection Point - INAP Operation. This is the operation that is sent when the switch reaches a trigger detection point.

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

ISDN
Integrated Services Digital Network - set of protocols for connecting ISDN stations.

ISUP
ISDN User Part - part of the SS7 protocol layer and used in the setting up, management, and release of trunks that carry voice and data between calling and called parties.

ITU
International Telecommunication Union

M3UA
MTP3 User Adaptation. The equivalent of MTP in the SIGTRAN suite.

MAP
Mobile Application Part - a protocol which enables real time communication between nodes in a mobile cellular network. A typical usage of the protocol would be for the transfer of location information from the VLR to the HLR.

MS
Mobile Station

MSC
Mobile Switching Centre. Also known as a switch.

MT
Mobile Terminated
**MTP**
Message Transfer Part (part of the SS7 protocol stack).

**MTP3**
Message Transfer Part - Level 3.

**NOA**
Nature Of Address - a classification to determine in what realm (Local, National or International) a given phone number resides, for the purposes of routing and billing.

**NP**
Number Portability

**Oracle**
Oracle Corporation

**PC**
Point Code. The Point Code is the address of a switching point.

**Peer**
Remote machine, which for our purposes is capable of acting as a Diameter agent.

**PLMN**
Public Land Mobile Network

**SCCP**
Signalling Connection Control Part (part of the SS7 protocol stack).

**SCTP**
Stream Control Transmission Protocol. A transport-layer protocol analogous to the TCP or User Datagram Protocol (UDP). SCTP provides some similar services as TCP (reliable, in-sequence transport of messages with congestion control) but adds high availability.

**SGML**

**SGP**
Signalling Gateway Process.

**SGSN**
Serving GPRS Support Node
SLC
Service Logic Controller (formerly UAS).

SLEE
Service Logic Execution Environment

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

SN
Service Number

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).

SS7
A Common Channel Signalling system used in many modern telecoms networks that provides a suite of protocols which enables circuit and non circuit related information to be routed about and between networks. The main protocols include MTP, SCCP and ISUP.

SSN
Subsystem Number. An integer identifying applications on the SCCP layer.
For values, refer to 3GPP TS 23.003.

SSP
Service Switching Point

STP
Signalling Transfer Point. Telecom equipment routing SS7 signalling messages.

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.

USSD
Unstructured Supplementary Service Data - a feature in the GSM MAP protocol that can be used to provide subscriber functions such as Balance Query and Friends and Family Access.

VLR
Visitor Location Register - contains all subscriber data required for call handling and mobility management for mobile subscribers currently located in the area controlled by the VLR.
## Index

### A

- AAA • 61
- About Installation and Removal • 59
- About This Document • v
- ACS • 61
- activate • 31
- address indicator • 44
- alwayssendaddr • 13
- ansi • 22
- application_server • 32
- AS • 61
- ASN.1 • 61
- asn1_validate • 23
- ASP • 61
- asp_identifier • 32
- Audience • v
- autoac • 13
- Available TCAP Interfaces • 2

### B

- Background Processes • 53

### C

- CAMEL • 61
- CAP • 61
- Checking the Installation • 59
- Checklist - non-SMS • 60
- Checklist - SMS • 59
- Classifier parameters • 23, 41, 42, 44
- classifiers • 23
- Common Troubleshooting Procedures • 55
- Configuration • 9, 53, 54
- Configuration components • 9
- Configuration Overview • 3, 4, 9, 53, 54
- Configuration process overview • 10
- Configuring sigtran.config • 3, 4, 9, 20
- Configuring tcapif.def • 3, 4, 9, 12
- Connection • 61
- Connection key configuration • 31
- Connection parameters • 24, 31, 41
- connections • 23
- Copyright • ii
- Correlation IDs • 5, 7
- cron • 61
- crontab • 61

### D

- Debug • 55
- default_gt • 32
- default_retgt • 24
- Defining the parameters • 12
- defoutac • 14
- Diagram - route 2 PCs • 49
- Diagram - route all to balanced ITPs • 50
- Diagram - route on transmission type • 51
- Diameter • 61
displaymonitors • 14
- Document Conventions • vi
- DP • 62
dpause • 14
- DTMF • 62

### E

- Editing the file • 20
- Environmental variables • 9, 10
- ESERV_CONFIG_FILE • 11, 20
- Example - route 2 PCs • 48
- Example - route all to balanced ITPs • 49
- Example - route on transmission type • 50
- Example Configuration Scenarios • 48
- Example SLEE service keys • 6

### F

- first • 42

### G

- GPRS • 62
- GSM • 62
- GT • 62
gti • 45
gtt_np • 33
gtt_pc • 33
gtt_remove • 33
gtt_route_pc • 33
gtt_ssn • 34

### H

- HLR • 62
- HPLMN • 62
- HTML • 62

### I

- IDP • 62
- importance • 34
- IN • 63
- INAP • 63
- inapssns • 14
- Initial DP • 63
- initiation • 34
- Installation and Removal Overview • 59
- Introduction • 1, 2, 3, 5, 9, 20, 55, 57, 59
- Introduction to SLEE TCAP Interfaces • 1
- invokeTimerOverride • 26
- invokeTimers • 25
- IP • 63
- IP address • 63
- ISDN • 63
- ISUP • 63
- ITU • 63
SIGTRAN_CONFIG_SECTION • 11
SLC • 65
SLEE • 65
SLEE Correlation ID diagram • 7
SLEE service key construction • 6
SLEE_ETC_DIR • 11
sleekey • 6, 18
SMS • 65
SN • 65
sndbuf • 40
SRF • 65
SS7 • 65
SSN • 65
ssns • 18
SSP • 65
Startup • 53, 54
Statistics • 57
Statistics and Reports • 57
Statistics for all connections • 57
statisticsInterval • 30
statsf • 19
stderron • 19
STP • 65
stpPCs • 30
stps • 19
SUA • 65
SUA connections • 58
sua_if • 3, 53
subsystem_number • 46
System Administrator • 65
System Overview • 1
T
TCAP • 65
TCAP Interface layers • 2
TCAP Protocol stack • 1
tcapif parameters • 5, 13
tcapif.def • 12
TCAPIF_DEF • 11
TCP • 66
traffic_mode_type • 40
trans_type • 47
transport • 41
Troubleshooting • 55
Typographical Conventions • vi
U
use • 41, 43
Using eserv.config instead of sigtran.config • 20
USSD • 66
V
VLR • 66
X
xudtHopCount • 30